A Linear Combination of Heuristics Approach to Spatial Sampling Hyperspectral Data for Target Tracking

DISSERTATION

Barry R. Secrest, Major, USAF

AFIT/DEE/ENG/10-08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

Disclaimer Statement: The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

Copyright Declaration: This material is declared a work of the United States Government and is not subject to copyright protection in the United States.

AFIT/DEE/ENG/10-08

A LINEAR COMBINATION OF HEURISTICS APPROACH TO SPATIAL SAMPLING HYPERSPECTRAL DATA FOR TARGET TRACKING

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Barry R. Secrest, BS, MS

Major, USAF

December 2010

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

AFIT/DEE/ENG/10-08

A LINEAR COMBINATION OF HEURISTICS APPROACH TO SPATIAL SAMPLING HYPERSPECTRAL DATA FOR TARGET TRACKING

Barry R. Secrest, B.S.E.E., M.S.E.E.
Major, USAF

Approved:

//SIGNED//	23 June 2010
Lt Col (ret.) Juan R. Vasquez, PhD (Chairman)	Date
//SIGNED//	23 June 2010
Dr. Kenneth Bauer, (Member)	Date
//SIGNED//	23 June 2010
Major Michael J. Mendenhall, PhD (Member)	Date

Accepted:

//SIGNED//	13 July 2010
M. U. Thomas Dean, Graduate School of Engineering and Management	Date

AFIT/DEE/ENG/10-08

Abstract

Persistent surveillance of the battlespace results in better battlespace awareness which aids in obtaining air superiority, winning battles, and saving friendly lives. Although hyperspectral imagery (HSI) data has proven useful for discriminating targets, it presents many challenges as a tool in persistent surveillance. HSI sensors gather data at a relatively slow rate yet the sheer volume of data can be overwhelming. Additionally, fusing HSI data with grayscale video is challenging due to the differences in frame rate and resolution. A new sensor under development has the potential of overcoming these challenges and transforming our persistent surveillance capability by providing HSI data for a limited number of pixels and grayscale video for the remainder.

This dissertation explores the exploitation of this new sensor for target tracking. Every pixel receives a utility function value based on nearness to a target of interest (TOI) (determined from the tracking algorithm) and components of the TOI. The components in the utility function are equal dispersion, periodic poling, missed measurements, and predictive probability of association error (PPAE) which is introduced as a statistical measure in this dissertation. Equal dispersion means each TOI receives an equal amount of the resources available. Periodic poling means resources are dispersed based on how long it has been since the TOI received an HSI update. TOIs that were recently updated will receive fewer resources, while those that have not had an update for a longer time will receive more resources. The missed measurement component gives more resources to TOIs that have missed measurements. The more measurements a TOI has missed, the more resources it will get. PPAE is the probability that a TOI will receive measurements that result in an association error. It is predictive since the probability can be calculated at any time in advance of receiving measurements. It varies with nearness to other TOIs and the nature of their covariance matrices. PPAE is theoretically derived and validated in experiments.

Experiments are conducted to validate the individual components. Experiments use a simulated urban environment and a Kalman filter multitarget tracking algorithm to compare the individual components to each other. The synergism of the utility function which uses all the components is shown to outperform all individual components and is 6.5 percentage points better than the baseline performance of equal dispersion. The new sensor is successfully exploited resulting in improved persistent surveillance.

Table of Contents

List of Figures

List of Tables

A LINEAR COMBINATION OF HEURISTICS APPROACH TO SPATIAL SAMPLING HYPERSPECTRAL DATA FOR TARGET TRACKING

1 Introduction

Perhaps the most drastic difference in battlespace awareness facing the U.S. military in Iraq has been the need to conduct military operations in an urban environment. Persistent surveillance in an urban environment has been credited with success in Iraq [1]. Although we have had the ability to track large numbers of targets with various platforms and sensors for over a decade, the urban environment requires us to positively identify enemy targets and separate them from non-combatants. Previous battlespaces have taken a friend-or-foe approach. If it is not a friend, it must be a foe. The introduction of a third category, the non-combatant, requires a new paradigm. This dissertation enables greatly improved target identification in an urban environment.

1.1 Persistent Surveillance Using a Single Platform and Sensor

Persistent surveillance of a battlespace has been an Air Force goal for the past two decades [2]. A result of our persistent surveillance in Fallujah, Iraq allowed us to know where the enemy was better than their own leadership and resulted in saving American and allied lives [1]. Improving technologies for persistent surveillance is a recognized need of the Air Force in order to overcome challenges of complex environments and tighter rules of engagement needed to combat terrorists amid non-combatants [3]. Persistent surveillance can be accomplished using a single platform such as Predator or Global Hawk. Panchromatic video or hyperspectral imagery from a single moving platform is gathered from a sufficient height using a fairly large field of view (FOV) and is stabilized, geolocated, oriented, and stitched together to form an overhead picture of the urban environment. The end result is a large single over-head video of the battlespace. The technical challenges of tracking enemy targets using persistent surveillance include the challenges with producing the data as a single, stable, overhead video, the typical challenges to tracking using video or hyperspectral images, and additional challenges presented by advanced sensor capabilities requiring a resource manager.

1.1.1 Challenges of Producing Persistent Surveillance Data

Using a moving platform to produce an over-head video that appears to have been created from a single stationary over-head platform is challenging. The challenges include frame-referencing, image stitching, and geolocation. The goal of frame-referencing is to remove apparent motion caused from the motion of the platform or sensor. The video of a stationary camera mounted on an overhead plane flying in a straight line would show buildings apparently moving. Frame-referencing or image registration would establish a reference frame and compare adjacent frames to it to remove the apparent motion caused from the platform while showing the true relative motion of vehicles or targets [4]. Image stitching merges one or more images into a single image [5]. Adjacent frames of video from a moving platform have portions which overlap and portions which are new, which are then stitched together to form a single cohesive image. Geolocation is the process of turning image information into precise coordinates [6]. This step is vital for handing the information to an external process such as targeting by a weapons platform.

The errors or anomalies that exist in the persistent data as a consequence of not completely overcoming the challenges become target tracking challenges.

1.1.2 Challenges of Target Tracking Using Panchromatic Video

The challenges of target tracking using panchromatic video include segmentation and some relatively common challenges to tracking in general such as measurement noise, association errors, clutter measurements, and missing measurements. Segmentation is the process of producing measurements from the sensor output [7]. Errors in persistent surveillance data compound the challenge of segmentation. Segmentation errors in turn show up as general target tracking errors. Measurement noise is the result of an imprecise measurement. Association errors occur when two targets are close enough together and the measurement noise is large enough that the measurements of the targets are confused and associated with the wrong track. The segmentation process can produce spurious or clutter measurements which are false measurements that either do not originate from a target (spurious) or are extra measurements resulting from a mirror of the target (clutter).

1.1.3 Challenges of Target Tracking Using Hyperspectral Imagery

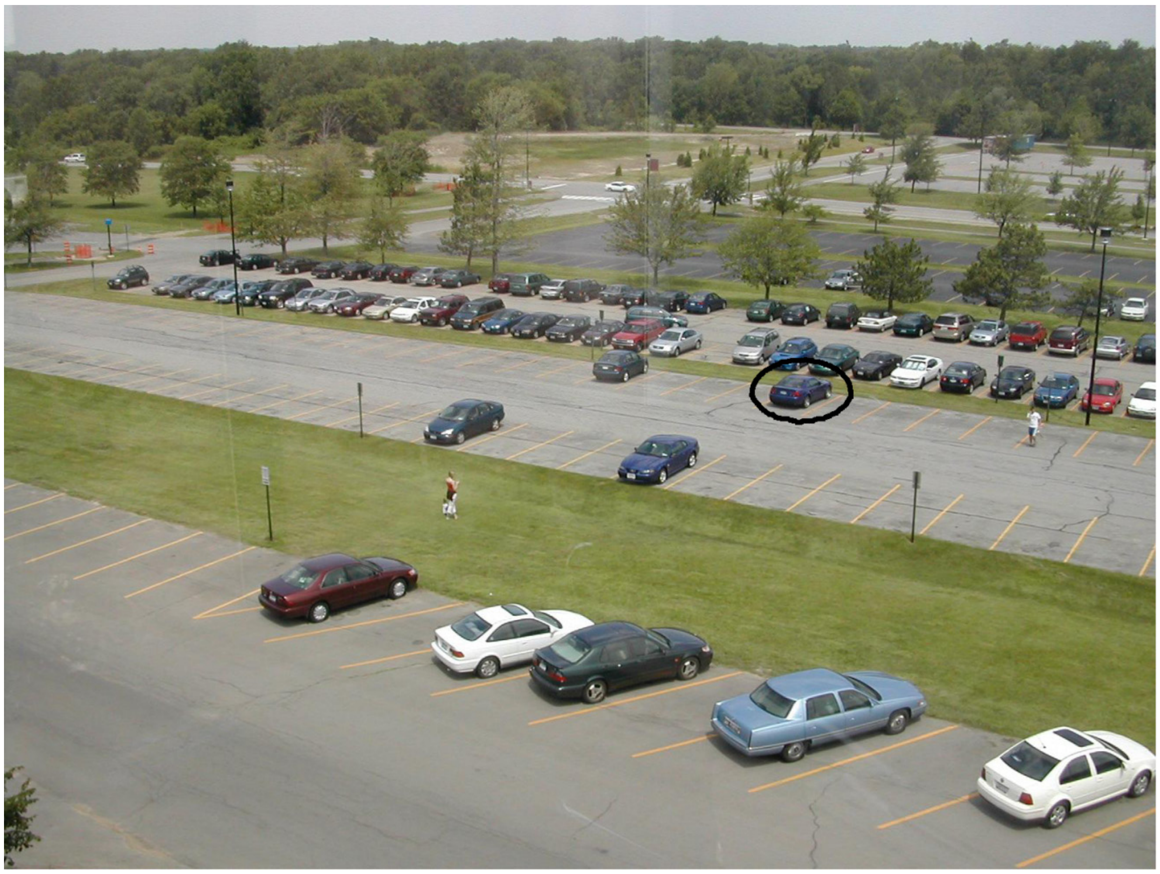


Figure 1 - Four blue cars in a parking lot. These cars are similar in color and are difficult to tell apart from a distance.

Imagine our goal is to keep track of the circled car from Figure 1. This car and the other blue cars around it will drive around. Sometimes they will be quite near each other and sometimes they will be out of sight for short durations of time. When they come back into view, we need to figure out which one is the circled car. The particular shade of blue of our car should let us keep track of it. Now imagine we need to keep track of the blue car using an airborne surveillance system. Figure 2 shows these same blue cars from

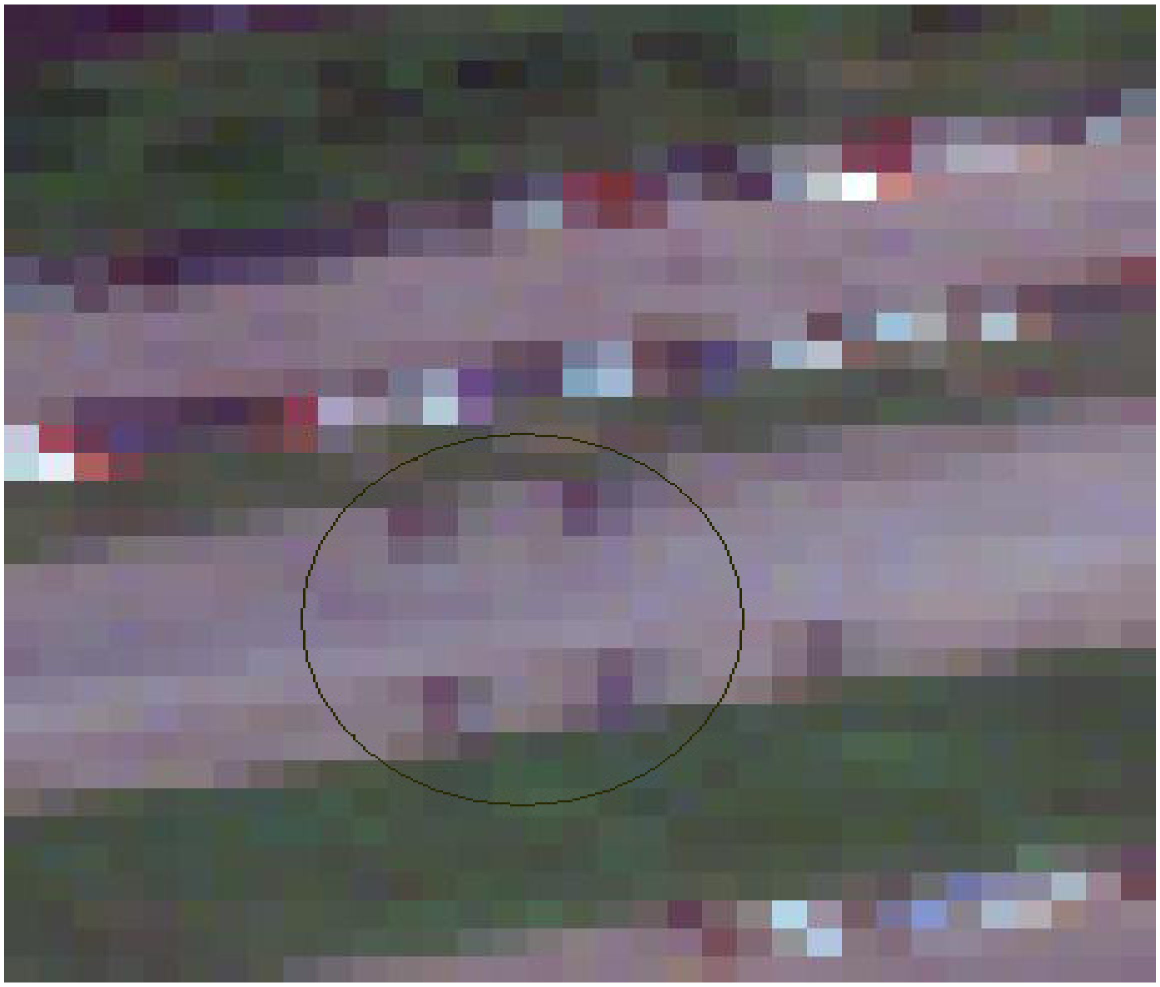


Figure 2 - The same four blue cars from Figure 1 viewed from 2,500 ft.

2,500 ft using a very large field of view (FOV) image. The lower spatial resolution makes it very challenging to distinguish the cars from one another. By using hyperspectral imagery (HSI), a pixel from the blue car in Figure 1 was found and identified in the image corresponding to Figure 2 [8]. Hyperspectral imagery provides the ability to positively identify vehicles and thus overcome one of the challenges of target tracking.

Although hyperspectral imagery (HSI) has been shown to reduce association errors and increase target identification, it has some additional challenges in framerate and data size. Paradoxically, it is too much data and we cannot get enough of it. Although improvement in sensor technology will increase the framerate in the future, current technology for the desired pixel size and field of view (FOV) allow a framerate of approximately 1 frame per 10 seconds. Since panchromatic video has a framerate of about 30 frames per second, a purely panchromatic target tracker will generally outperform a hyperspectral target tracker. Additionally, a frame of HSI contains up to 200 times the data contained in a single frame of panchromatic video. This creates problems with transmission, storage, and computational complexity. The two main approaches of reducing the data are spectrally and spatially. Spectral data reduction reduces the amount of spectral data for every pixel by eliminating spectral bands gathered or transmitted. Spatial data reduction selectively gathers or transmits only a subset of pixels from the entire hyperspectral image.

1.1.4 Challenges with Multi-Modal Sensors and Resource Managers

One method of overcoming challenges associated with either panchromatic video or hyperspectral imagery is to use a multi-modal sensor. In panchromatic video, the camera could zoom in on a specific target to aid in identification or to help refine the measurement and avoid association errors. The video verification of identity (VIVID) program tracked multiple targets by rapidly slewing a single camera such as on a Predator [9]. The challenges of this approach are accounting for the physical limitations of the device such as slew time and settling time between targets, focusing time for zooming, as well as spatially determining the correct location to pan to. A resource manager is used to maximize tracking performance by sending the appropriate pan/tilt/zoom commands to the sensor. Tracking performance in the VIVID system was limited by the physical constraints of the slewing platform[9].

Soliman uses a hyperspectral sensor with a pushbroom approach for image acquisition and applies it to the vehicular tracking problem. This sensor has the capability of selectively obtaining a single line of pixels in the image. Again, the resource manager had to deal with the challenges of settling time and a cost (in time delay) of selecting an arbitrary line rather than the next line. Additionally, this approach had challenges associated with the image temporal disparity since individual adjacent lines could be collected temporally out of order and not necessarily temporally adjacent. This added complexity to image stitching [10].

1.2 Persistent Surveillance Using Multiple Platforms and Sensors

In order to overcome some of the challenges of single platform persistent surveillance, multiple platforms and sensors are used. Wang proposes a single platform but with both video and HSI sensors and demonstrates how they can be used together for target tracking, however the resolution and FOV were such that thousands of pixels were captured for the target and are impractical for a persistent surveillance application which requires a much larger FOV [11]. While it is possible to have a single platform with multiple sensors, it is more common for multiple platforms such as a Predator and a Global Hawk to both be simultaneously surveillancing an area of interest. Besides all of the challenges that each platform must face individually, there are two types of challenges that are unique to multiple platforms. These challenges are the spatial constraints presented to a resource manager by the various platforms and the challenge of data fusion.

1.2.1 The Challenge of a Spatial Resource Manager

While each individual platform may have a resource manager to manage the multiple modes available by the sensor, a multi-platform system needs an overall spatial resource manager. The spatial resource manager decomposes the tracking problem by allocating areas or targets that need to be observed by each individual platform. Since each platform may have different capabilities, the spatial resource manager needs to assign the platforms in such a way to maximize tracking performance. While it may make sense to have a central system controlling multiple platforms, our acquisition process makes controlling multiple platforms such as Predator and Global Hawk from a single control system rather difficult.

1.2.2 The Challenge of Data Fusion

Since multiple platform surveillance is generally adhoc and accomplished without a resource manager, data fusion is a more logical solution. Data fusion is the process of combining data to refine state estimates and predictions for the targets being tracked [12]. Data fusion can occur at several points within the tracking process. With a Predator and Global Hawk both collecting panchromatic video, data fusion can occur for the separate images collected to produce a single image. This form of data fusion has the challenges of image stitching and frame referencing which were discussed in Section 1.1.1 and is additionally made more complex due to potential differences in video resolution, time differences due to differing framerates and transmission delay times, and of course perception differences due to platform location disparity. If the sensors are not compatible, such as radar and panchromatic video, data fusion must occur later in the process. Measurements from both sensors can be fused into a single measurement list. The differences in measurement rates and timing likewise are a challenge to this process. In addition, the accuracy and measurement noise from each system needs to be accurately modeled in order to effectively combine the measurements. Finally, data fusion can occur after the individual measurements have been processed by each independent system, thus the track information from each system can be fused. Again, the differences in measurement rates and timing will present a challenge since the data will need to be fused after measurements are used by each system. In addition the tracking model for and accuracy for each system will need to be known in order to fuse the data.

1.3 The New Multi-object Spectrometer

The new multi-object spectrometer (MOS) is designed to overcome or avoid many of the challenges of persistent surveillance by selectively gathering HSI data for a limited number of pixels (see Section 9.2 for a description of the MOS). It can be used either as a single platform, or in combination with other platforms. It helps overcome some of the challenges of target tracking using panchromatic video by augmenting the data with HSI. It allows a higher framerate with lower volume of data collected by spatially reducing the data and selectively gathering HSI data. Since the HSI is obtained with the video, there are no timing or video resolution challenges normally associated with multi-modal sensors or data fusion. The challenge of exploiting this new sensor is in developing a spatial resource manager to take advantage of the unique capabilities of the sensor.

1.4 Research Scope

This research develops a spatial resource manager for the new MOS sensor when used for target tracking. Spatial sampling is solved using a utility function where pixels receive a value based on their nearness to a target of interest (TOI) and some determining characteristics of the specific TOI. Although prior resource managers have used a utility function and spatial track state information, this work presents unique heuristics and additionally combines them as a linear combination in the utility function. Although these heuristics were developed specifically for the MOS sensor, they are generally applicable to spatial resource managers for target tracking.

The TOI receives a base value from a linear combination of its determining characteristics. The maximal pixel value is the base TOI value and occurs when the TOI estimated position is at the pixel. As pixels get farther from the estimated position of the TOI, the pixel value decreases. The minimum pixel value is zero. The TOIs are determined from the tracking algorithm, thus providing a close coupling of the tracking and the sensor control. The components or heuristics in the utility function are equal dispersion, periodic poling, missed measurements, and predictive probability of association error (PPAE). The last three heuristics are novel ideas presented in this dissertation. Equal dispersion means each TOI receives an equal amount of the resources available and is a base value for the existence of any TOI. Periodic poling means resources are dispersed based on how long it has been since the TOI received an HSI update. TOIs that were recently updated will receive fewer resources, while those that have not had an update for a longer time will receive more resources. The missed measurement component gives more resources to TOIs that have missed measurements. The more measurements a TOI has missed, the more resources it will get. Finally, PPAE is introduced in this dissertation. PPAE is a statistical measure of the probability an association error will occur. PPAE is first theoretically derived for the one dimensional case with two targets. It is validated by an experiment that approximates the PPAE by randomly generating measurements for two targets and comparing the result to the calculated PPAE. The one dimensional case is then extended to two dimensions and a similar validation experiment is conducted by randomly generating measurements for two targets to approximate the PPAE and compare it to the calculated PPAE. The final validation experiment for PPAE occurs inside a tracker and compares the predicted results to actual association errors obtained on realistic multi-target simulation data

The individual heuristics of the utility function are tested using simulated HSI and solving the spatial sampling problem. Experiments are conducted to validate the individual components and compare them to each other. These experiments show that equal dispersion is the poorest individual component followed closely by missed measurements. PPAE performs better than equal dispersion and periodic poling is the best individual component. The synergism of the utility function is shown through equally weighting the components and is shown to outperform periodic poling. The relative importance or optimal weighting of the different types of TOI is accomplished by a genetic algorithm using a multi-objective problem formulation. This optimized solution slightly outperforms equally weighting the utility function and illustrates the near optimal performance of equal weighting.

1.5 Organization

Though solving spatial sampling is the goal of this dissertation, there are several byproducts which are meaningful. These byproducts are 1) a simulator tool for the creation of simulated vehicle traffic, 2) tuning the multi-target tracker with a multi-objective function, 3) demonstrating Monte Carlo simulation may not be needed for tuning a multi-target tracker, 4) a tool for the new sensor design tradeoff analysis; analyzing performance space of framerate, probability of correct classification, and number of pixels to be gathered, 5) calculating the probability a target is in an arbitrary region, 6) calculating the joint probability two targets are in the same arbitrary region, and 7) PPAE as a useful statistical measure. The chapters are organized to effectively present these meaningful contributions. Related work and background information is presented in Chapter 2. The probability a target is in an arbitrary region, the joint probability of two targets in an arbitrary region, and the PPAE along with all the components of the utility function are discussed in length in Chapter 3 - problem formulation and proposed solution. The simulator tool is described in Chapter 4 - implementation details; which lays the foundation for understanding the experiments. Tuning the tracker and demonstrating a lack of need for Monte Carlo simulation is performed in Chapter 5 - design of experiments. Results, analysis, conclusions, and future work are presented in Chapter 6. Likewise the synergy of the utility function is shown by comparing it to each individual component or heuristic of the utility function in the same Chapter. Finally, Chapter 7 gives a summary of all of the contributions.

2 Related Work and Background

2.1 Related Work

Broadly speaking this work uses HSI to perform target tracking. More specifically, it presents a resource manager that performs a data reduction by doing spatial subset selection using targets of interest (TOI). Related works are in the intersecting areas of target tracking, sensor resource management (SRM), HSI data reduction, and optimization or tracking heuristics.

2.1.1 Resource Managers for Panchromatic Video

The SRM problem is formally presented by Washburn [13]. Washburn uses a utility function to determine which TOI the sensor should obtain a measurement from however the specific heuristics differ and are discussed in Section 2.1.2. The integrated sensing and processing (ISP) program studied the SRM problem and is applicable generally, producing upper and lower bounds on SRM performance [14]. The ISP SRM is a closed loop system between the sensor and the fusion system and is not as closely coupled with the tracking algorithm as in this work. The VIVD program's SRM models the effect of sensor observations on tracking performance to determine the optimal desired observations, however sufficient detail is not provided to understand their heuristics [9].

2.1.2 Tracking Heuristics for use in Resource Managers

Washburn's utility function uses two heuristics (detection probability and association uncertainty) that serve the same purpose as two of the heuristics presented in this work (missed measurements (MM) and probability of association error (PPAE)) [13]. Washburn models the detection probability for each TOI. The purpose of the detection probability is to model the effects of obscuration where an attempted observation is less likely to succeed. The detection probability is determined from context mapping and past attempts within regions. Washburn uses the detection probability to lower the utility in areas of obscuration. In contrast, the MM heuristic as presented in this work increases the utility in areas of obscuration since HSI measurements have a higher probability of detection than panchromatic video. In addition, MM is not calculated from context mapping, but is a state of each track maintained by the tracker based on past missed measurements. Washburn's association uncertainty uses the track uncertainty and the expected number of confusers to heuristically approximate the PPAE. PPAE differs in that it is a calculated probability which also includes the track uncertainty for each confuser as well as the relative position among them. While Washburn, ISP, and VIVID all use the track estimated position, none of these RSMs are concerned with the utility of positions near the estimated position. This difference is due to the fact that the MOS sensor needs a pixel-level RSM and the other RSMs need only point the camera toward the region of the estimated position.

2.1.3 Optimization Using a Linear Combination of Heuristics

The idea of combining several individual heuristics together is not new and is known as hyper-heuristics or heuristics to choose heuristics [15]. In a typical problem, a solution is arrived at by repetitively calling the heuristic at each decision point. In hyper-heuristics, a different heuristic is selected to be called at each decision point. Most NP-complete problems, such as the job-shop problem, have deterministic heuristics that do not lend themselves to numeric linear combination [16]. There are no linear combinations of heuristics in the literature.

2.1.4 Hyperspectral Imaging and Target Tracking

HSI features have been used to track moving targets in backgrounds involving both clutter and noise. HSI features have proven to be superior to those of standard imagery [17]. Specific areas of relevance to the Air Force (AF) include pure tracking using HSI, using HSI feature data, and doing HSI fusion with other sensor types [18]. Soliman demonstrates a HSI-Augmented panchromatic video target tracker [10].

2.1.5 Hyperspectral Imaging Data Reduction

Since HSI data for a single image contains much more information than standard video, HSI data presents challenges to current technology both for storage of information and transmittal over ground or satellite-based networks and is a relevant AF problem [19]. HSI data reduction often uses a transformation matrix to change the high-dimensionally HSI data to a lower dimension while preserving the ability of a classifier to perform well. Data reduction can be further broken into feature subset selection which removes spectral data from each pixel and spatial subset selection which removes pixels from the image.

HSI feature subset selection is a form of HSI data reduction. Generally speaking, HSI data reduction requires all the original data in order to perform the reduction since the reduced data is a linear transformation of the original data. While this may help with data storage and transmittal issues, it still means that all the data must be gathered for subsequent data reduction and classification. Feature subset selection produces a subset of data from the original. The goals of feature subset selection are to minimize the number of features in the subset while maximizing the resulting probability of correct classification. Genetic algorithms have been used to perform feature subset selection [20], [21]. While feature subset selection is a linear transformation like HSI data reduction, the transformation can be represented by the identity matrix with zeros on the diagonal corresponding with data to be removed. Since the remaining data has no linear relationship with the removed data, we no longer need to collect samples of the removed data to produce reduced data. Feature subset selection does not require all the original data the same way that generalized HSI data reduction does. (See Chapter 10 to see how the advanced multi-object spectrometer (MOS) sensor can be used for feature subset selection which is beyond the scope of this work.)

Spatial subset selection is more relevant to this work. Although there are several works regarding spatial subset selection in the literature, methods that rely on a region of interest or TOI for selection are application specific. Spatial subset selection is relevant to target tracking is captured above in Section 2.1.2.

2.2 Background

This work uses current optimization techniques and develops new techniques applied to the discipline of target tracking. Since it is a multi-disciplinary work, we have chosen to provide optimization examples and gear the language toward the target tracking discipline.

We describe current target tracking algorithms including segmentation, scoring and gating, association, Kalman filtering methods, and track confirmation and deletion. We further describe optimization terms such as optimization problem, tuning parameters, adaptive tuning parameters, heuristic decision makers, and evolutionary algorithms. Finally, since this work makes use of HSI, we include information relevant to HSI including the HSI classifier. The purpose is to provide sufficient background to familiarize the reader with important terminology needed to understand later sections.

2.2.1 Models and States

A model is a mathematical representation of a real-world physical system. It is descriptive, predictive, and approximate of the real system [22]. The two models used in this work to represent the two-dimensional (2D) dynamic motion of vehicles in traffic are the white noise constant velocity (CV) dynamic model and the white noise constant acceleration (CA) dynamic model. The CA model is more complex and may be more precise than the CV model for more dynamic targets and is used to create simulated traffic data. The CV model is used in the Kalman filter of the tracker to predict track positions instead of the CA model to reduce computational complexity and to correspond with the design objective of providing a lower performance baseline as discussed in Section 5.11.

States are the variables used in the mathematical model and additional variables of interest which may be derived from the model. The symbol **x** is used to denote the state vector. In the case of the CV model, the states are the position *x* and the velocity *v*. With a 2D model, both the position and velocity may be broken into separate 2D components based on the Cartesian coordinate system. (i.e., x_x and x_y position and v_x and v_y velocity).

2.2.1.1 White Noise Constant Velocity Dynamic Model

If an object were travelling in a frictionless environment with no acceleration, future positions could be predicted using well-known physics equations from the current position, the velocity, and the change in time. If instead of no acceleration, vehicle acceleration is modeled as white noise we have

$$\dot{\mathrm{x}} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w(t) \tag{1}$$

$$a(t) = w(t) \tag{2}$$

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{3}$$

where a dot above a symbol represents the time derivative of the variable, **x** is the state variable comprised of *x* and *v*, *x* is the position, *v* is the velocity, *a*(*t*) is the acceleration at time *t*, Φ is the state transition matrix embodying the physical model, *T* is the discrete change in time, and *w*(*t*) is the white noise process defined by:

$$E[w(t)] = 0, \qquad E[w(t)w(\tau)] = q(t)\delta(t-\tau) \tag{4}$$

The resulting process noise covariance *Q* becomes [7]:

$$Q = \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{3} \\ \frac{T^2}{3} & T \end{bmatrix} q \tag{5}$$

where q is the strength of the model uncertainty.

2.2.1.2 White Noise Constant Acceleration Dynamic Model

This model is sometimes referred to as the white jerk model since a change in acceleration is jerk [7]. In this model we have a constant acceleration and the change in acceleration is modeled as white noise. It is similar to the CV model, but is of a higher order. Its equations are:

$$\dot{\mathrm{x}} = \begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{a} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w(t) \tag{6}$$

$$\dot{a}(t) = w(t) \tag{7}$$

$$\Phi = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} \frac{T^5}{20} & \frac{T^4}{8} & \frac{T^3}{6} \\ \frac{T^4}{8} & \frac{T^3}{3} & \frac{T^2}{2} \\ \frac{T^3}{6} & \frac{T^2}{2} & T \end{bmatrix} q \tag{8}$$

Both the CV and CA models can represent the same real-world system of dynamic motion. Since the CA model is of a higher order, it should produce more realistic and accurate predictions when the dynamic motion has significant changes in acceleration. However since it is more complex, the CA model requires greater computational power, and for the targets being modeled in this problem, the CV model is more than adequate.

2.2.2 Filtering

The Kalman filter is an iterative algorithm that uses a model to predict future states. It consists of a propagate cycle where the current state is propagated into the future and an update cycle where a measurement of the system is used to adjust the current state.

2.2.2.1 Update

If a track is associated with a measurement, that information is used to update the current estimate of the state variables of the track. An update always results in an increased certainty of the state variables. If there is no measurement, no update occurs. The update equations are [23]:

$$\begin{aligned} S &= HPH' + R \\ K &= PH'S^{-1} \\ \widehat{X} &= \widehat{X} + K(Z - H\widehat{X}) \\ P &= P - KHP \end{aligned} \tag{9}$$

where *X* is the state variable, *Z* is the measurement, *S* is the covariance of the innovation (*Z* - *HX*), *P* is the state covariance, *R* is the measurement noise covariance, *H* is the measurement matrix, *K* is the Kalman gain, and ^ above a variable denotes the value is estimated.

2.2.2.2 *Propagate*

Not all tracks receive a measurement update, but all tracks are propagated forward in time to the next frame of measurements. Propagation naturally results in a decreased certainty of the state variables since we are predicting the future based on the past. The propagate equations are [23]:

$$\begin{aligned} \widehat{X} &= \Phi \widehat{X} \\ P &= \Phi P \Phi' + Q_d \end{aligned} \tag{10}$$

where Φ is the state transition matrix which embodies the dynamic motion model and Q_d is the discrete time model uncertainty. Note that alternate forms for both the update and propagate equations exist and can be found in [23].

2.2.2.3 *Pre-Computation of the Steady-State Covariance and Innovation*

It is well known that for a given value of R and Q_d values of P can be pre-computed in the case of a linear Kalman filter and that P reaches a steady-state value (P_{SS}) [23]. The reason this is possible is because P in Equation (9) and Equation (10) does not depend on the measurement Z. P_{SS} is computed by iterating with Equation (9) and Equation (10) until the new value of P approximately equals the prior value of P. There is an inherent assumption that a measurement update is available every time.

Since P reaches a steady state (P_{ss}), S reaches a steady state (S_{ss}). From Equation (9):

$$S_{ss} = HP_{ss}H' + R \tag{11}$$

The steady-state covariance of the innovation is important in order to understand the behavior of a heuristic discussed later.

2.2.3 Target Tracking

A target tracking algorithm takes input from a sensor such as radar or video and provides state variables such as position and velocity for TOIs. The sub-algorithms within the target tracking algorithm are segmentation, gating and association, track initiation, filtering (see Section 2.2.2), and confirmation and deletion. These will be explained in more detail below and Figure 3 illustrates the flowpath of a target tracking algorithm.

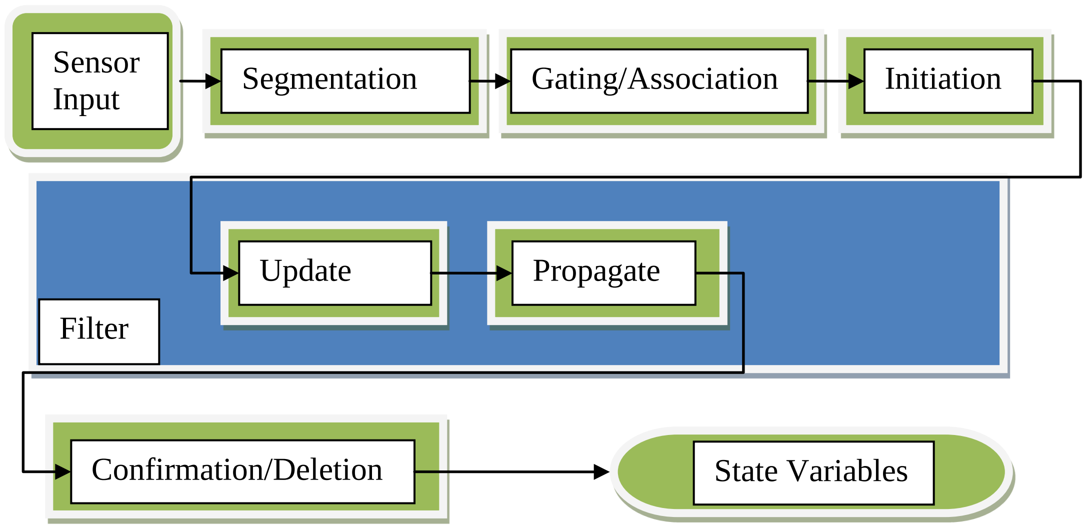


Figure 3 - A Typical tracking algorithm flowpath.

2.2.3.1 Segmentation

The sensor takes readings which the segmentation algorithm interprets to produce a list of target measurements [7]. Examples of motion segmentation algorithms that work with moving targets in optical data include frame differencing [24] and optical flow [25]. In frame differencing, each pixel in a frame is compared with the same pixel from one or more prior frames. If the intensity of the pixel changes drastically it is likely due to the motion of the target, thus those pixels become measurements. In optical flow the problem being solved is “What would each pixel of a prior frame need to do to create the current frame?” The pixels that move are then measurements of targets. Optical flow also produces a measurement of velocity of the target; however this information is likewise inherent in frame differencing and is calculated by the filter.

There are many other segmentation algorithms. Each algorithm provides differing measurements and has differing computational and accuracy performance characteristics. Generally speaking, the more accurate the segmentation is, the better the performance is of all downstream algorithms.

2.2.3.2 Association and Gating

The association algorithm matches the measurements to the existing tracks [7]. The Mahalanobis distance is usually the prime measure driving the algorithm [26]. The association that minimizes this total summed distance is the most probable association solution. The nearest neighbor (NN) heuristic association method finds the closest track to each measurement (in some order) and associates them [27]. It generally produces a good but sub-optimal solution. Auctioning is another association algorithm that allows all tracks to bid on measurements [28]. This method allows more flexibility and generally outperforms nearest neighbor.

Gating sets a threshold around each track for which an associated measurement must lie. The threshold can be a set distance (square box), or based on the Mahalanobis distance (elliptical) [7]. If a measurement does not fall within the gate of a track, the measurement will not be associated with that track.

2.2.3.2.1 Mahalanobis Distance and Gating

The Mahalanobis distance is a measure of the distance between a measurement or some other point and the estimate of the track [7]. Unlike Euclidean distance, it is a chi-squared value that relates a distance to the probability of the measurement being within that distance as explained in Section 2.2.5.4. It differs by taking the standard deviations of the various axes into account. It equals the square of Euclidean distance when all the standard deviations are equal to one and there is no cross correlation. To obtain the distance between the points *i* and *j* we have:

$$\chi_{ij} = r_{ij}^T S^{-1} r_{ij} \tag{12}$$

where χ_{ij} is the Mahalanobis distance between the points *i* and *j*, r_{ij} is the residual between the points *i* and *j*, and *S* is the innovation covariance from Equation (9).

Since the Mahalanobis distance is a chi-squared variable and represents a probability of being within the distance, it is used as a gate for the association routine. Measurements that are close enough to the track estimate are considered for association, while measurements that fall outside the gating distance are considered as too improbable and are rejected. A typical gate may be such that 99% of measurements are expected to be within the gate of the track. Outliers that are true measurements of the target certainly do happen, but are not associated with the track.

Mahalanobis gating is sometimes referred to as elliptical gating. The area within a gating distance of a track is seen to be an ellipse or ellipsoid (for greater than two dimensions).

2.2.3.2.2 Association is NP-Complete – Outline of Proof

The goal of association is to minimize the total Mahalanobis distance from tracks to measurements. This is NP-complete and can be shown to be equivalent to the traveling salesman problem, a well-known NP-complete problem [29]. In the travelling salesman problem, the goal is to find a connected path (that is to be travelled by a salesman) between points so that the total distance is minimized. Given a set of tracks and an equal number of measurements that are to be associated, we force a path to go alternately between measurement and track by setting the distance between any two tracks or between any two measurements to infinity. We set the distance from the track to each measurement as the Mahalanobis distance (Equation (12)). Finally, we set the distance from a measurement to any track to be zero. In this way, the association is created between track and measurement and the total Mahalanobis distance is minimized. A track will not be associated with another track since the distance between them is infinity (which will never be a minimum). After an association between track and measurement, the node after the measurement is forced to be a track because once again the distance between any two measurements is infinity. Any track can follow a measurement, and hence the distance to all is equally zero. Minimize the distance of this forced alternating path between tracks and measurements as per the travelling salesman problem, and we have solved the association problem.

The minimum summed Mahalanobis distance is the association solution that is most probable. Unlike the travelling salesman problem where the minimum distance is the answer, finding the optimal solution to the association problem can still result in having the incorrect solution. The most probable solution may not be the right one.

2.2.3.2.3 Nearest Neighbor

The nearest neighbor (NN) is a greedy algorithm used for association [27]. Each track is associated with the closest unassociated measurement. The order the tracks are associated is important since once a measurement is associated, it is unavailable for tracks lower in the order. While computationally simple, the NN solution for more complex problems with closely spaced targets is inferior to methods that take into account multiple track-to-measurement pairings simultaneously.

2.2.3.2.4 Auctioning

Auctioning seeks to overcome the limitations of a greedy algorithm such as NN [28]. In using the NN approach, suppose the very first track has two nearly identical choices and chooses one that is slightly closer. Further suppose that the second track can also choose between the same two choices, but must choose the second measurement since the first track has already chosen. If the first measurement is significantly closer to the second track than the second measurement, a reduction in the total distance can be realized just by swapping the two associations and letting the second track have the first measurement and the first track taking the second measurement. Auctioning lets each track iteratively bid on each measurement with the highest bidder winning. In the case described above, the first track would bid evenly on both measurements while the second track would bid more on the closer measurement, thus winning it and favorably resolving the conflict.

2.2.3.2.5 Multi-Hypothesis Tracking

Multi-Hypothesis Tracking (MHT) is an enhancement to an association routine such as NN or auctioning. Rather than finding a single association solution, it produces a number of association solution hypotheses. For each hypothesis, the tracking solution (state variables for all tracked vehicles) is updated and propagated forward. Because of this, each hypothesis increases the computational burden of the tracker, so limits are placed on the number of hypotheses that can be processed. Breadth and depth of a MHT also play a large role in determining the computational burden. Breadth is the number of hypotheses allowed to be generated at a given time. Depth is the amount of propagations that a hypothesis is allowed to be carried forward. The breadth raised to the depth power is the total maximum number of possible hypotheses allowed. In order to reduce the number of hypotheses, only the most probable hypotheses are retained. See Section 2.2.4 for a description and example on how the probability that a hypothesis is correct is calculated.

2.2.3.3 Initiation

If a measurement is not associated with any of the currently existing tracks, a new track is created. Unless there is some a priori knowledge of the state of the track, the measurement itself provides the best information for the initialization. In the CV and CA model, the initial position can come from the first measurement, however the initial velocity or acceleration need more measurements to properly initialize since they come from a change in position or velocity. Similarly, the initial state covariance matrix is set to a relatively small value unless there is some a priori knowledge such as the sensor measurement error. One way to overcome the lack of a priori knowledge about the initial velocity and acceleration is to artificially inflate the initial covariance matrix (thus increasing the association gate size) to account the uncertainty in velocity and acceleration. Since new tracks are sometimes the result of sporadic measurements caused from clutter, these new tracks are differentiated from confirmed tracks and are sometimes called unconfirmed tracks.

2.2.3.4 Confirmation and Deletion

A confirmed track is one in which we have high confidence in the existence of a valid target. This occurs after several measurements have been obtained for the target. The methods used to confirm a track vary, but all the methods reflect measuring the confidence. The two most prominent methods are *M*/*N* [30] and likelihood scoring [31]. *M/N* is the simplest and states if *M* updates occur out of the past *N* measurement times, then the track is confirmed. Since the state covariance decreases with each update and increases with each propagate, *M/N* creates a required confidence level (which can be calculated without regard to the measurements, see Section 2.2.2.3) for a confirmed track. In contrast, the likelihood method more directly measures the confidence level and uses a threshold to determine a confirmed track. The kinematic likelihood ratio assumes a Gaussian distribution for true target returns and a uniform distribution for false alarms and is defined by:

$$LR_K = \frac{p(D_K|H_1)}{p(D_K|H_0)} = \frac{V_C \; e(-d^2 / 2)}{(2\pi)^{M/2}\sqrt{|S|}} \tag{13}$$

where LR_K is the kinematic likelihood ratio, D_K is the data such as position measurements, H_1 is the true target hypothesis, H_0 is the false alarm hypothesis, $p(D_K|H_i)$ is the probability density function evaluated with the received data under the assumption hypothesis H_i is true, V_C is the measurement volume element under which the distribution assumptions for true and false alarms is met, d^2 is the Mahalanobis distance from Equation (12), M is the dimension of the measurements, and S is the covariance of the innovation from Equation (9). The variables in the likelihood ratio are the Mahalanobis distance and the innovation covariance. As the innovation covariance decreases towards the steady state value, the likelihood ratio increases for the same Mahalanobis distance. However, for the same residual, a smaller innovation covariance results in an increased Mahalanobis distance and an overall decreased likelihood ratio. In order for a track to be confirmed, the innovation covariance and residual must both be sufficiently small. Once a track is confirmed, it remains so until it is deleted.

Deletion is often related to confirmation. If using M/N confirmation, deletion is usually M_d/N_d. For example, if we confirm when we get 7 associations out of the last 10 measurements, we might delete if we only get 3 or less associations out of the last 10 measurements. Similarly, we might confirm a track when the likelihood score is greater than L_c and delete it when it is less than L_d.

2.2.3.5 Tracking Metrics

A listing of many performance metrics used in the evaluation of multi-target tracking along with equations is provided in [32]. These metrics have been applied in a single-objective fashion through the use of weighted sums of the desired performance objectives [33]. Although there are many well-established objectives, the three we consider are the fraction of missed targets (FMT), identification errors (IDE), and association errors (AE).

2.2.3.5.1 Fraction of Missed Targets

Completeness is defined as the proportion of real objects that should be tracked as compared to those that are being tracked at each time point in a given scenario. The FMT is specifically defined as one minus the completeness:

$$FMT(t_{score}) = 1 - \frac{N_V(t_{score})}{N_{should\_track}(t_{score})} \tag{14}$$

Where $N_V(t_{score})$ is the number of unique valid tracks at time t_{score} and $N_{should\_track}(t_{score})$ is the number of real objects that should be tracked at time t_{score}. The average FMT over all time for a given scenario is the FMT for the scenario.

2.2.3.5.2 ID Errors

The inputs to the *association routine* at each time point are the noise corrupted measurements and the current state of items being tracked. The output is an association between the current track state and measurements to be used in the update cycle (see Equation (9)). Upon initiation, each track is given the enumerated ID of the target which generated the measurement. This truth information is used only for the calculation of this metric and is not used in any way in tracking the targets. In subsequent associations, if the track is associated with a measurement which does not match the initiation ID, then it is an ID error (IDE). The average IDE per vehicle over all Monte Carlo runs and all time is recorded as the IDE for the scenario as defined as:

$$\begin{aligned} IDE(i, t_{score}) &= \begin{Bmatrix} 0 \text{ if } ID(i, t_{score}) \;==\; ID(i, t_{init}) \\ 1 \text{ if } ID(i, t_{score}) \;<>\; ID(i, t_{init}) \end{Bmatrix} \\ IDE(t_{score}) &= \sum\nolimits_{T}^{i=1} IDE(i, t_{score}) / T \\ IDE(scenario) &= \sum\nolimits_{T;Time}^{i=1;\, j=1} IDE(i, j) / Time \end{aligned} \tag{15}$$

where $ID(i, t_{score})$ is the ID of confirmed track i at time t_{score}, similarly $ID(i, t_{init})$ is the ID of confirmed track i at time t_{init} and $ID(t_{score})$ is the summation of all assignment errors at time t_{score} divided by the total number of confirmed tracks T at time t_{score}. *IDE(scenario)* is a percentage metric reflecting the percentage of errors of all possible confirmed track associations.

2.2.3.5.3 Association Errors

An association error is like IDE in that it is related to the *association routine*. It compares the results of the *association routine* with the *assignment routine*. The *assignment routine* is essentially the same code as the *association routine*, but uses different inputs and it is used only for metric evaluation purposes. The inputs to the *assignment routine* are target truth and current track state whereas the inputs to the association routine are noise-corrupted measurements of target truth and current track state. Any difference between the *assignment* and the *association routines* is therefore caused by the noise corrupted data and position estimation error and is an association error. The average association error per vehicle over all Monte Carlo runs and all time is recorded as the association error for the scenario as defined as:

$$\begin{aligned} AE(T, t_{score}) &= \begin{cases} 0 \text{ if } \mathrm{Agn}(i,t_{score}) \ == \ \mathrm{Asc}(i,t_{score}) \\ 1 \text{ if } \mathrm{Agn}(i,t_{score}) \ <> \ \mathrm{Asc}(i,t_{score}) \end{cases} \\ AE(t_{score}) &= \textstyle\sum_{T}^{i=1} AE(i, t_{score}) / T \\ AE(scenario) &= \textstyle\sum_{Time}^{i=1} AE(i) / Time \end{aligned} \tag{16}$$

where $\mathrm{Agn}(i, t_{score})$ is the *assignment* of target confirmed track i at time t_{score}, similarly $\mathrm{Asc}(i, t_{score})$ is the *association* of target i at time t_{score}. The $AE(t_{score})$ is the summation of all assignment errors at time t_{score} divided by the total number of confirmed tracks T at time t_{score}. *AE(scenario)* is a percentage metric reflecting the percentage of errors of all possible confirmed track associations for a given scenario.

2.2.4 Calculating the Probability of Association Error

The probability of association error (PAE) is the probability that a specific association solution is correct. See Section 2.2.3.2.5 for a discussion on MHT and the motivation for finding this value to reduce computational burden. The PAE for the *k*th hypothesis is [34]:

$$PAE(H\_k) = 1 - (N(H\_k) / \sum_{j=1}^{m} N(H\_j)) \tag{17}$$

where *m* is the total number of hypotheses, and $N(H\_j)$ is the numerator for the second term of the *j*th hypothesis defined by:

$$N(H\_j) = \prod_{i=1}^{n} P\_j(\chi_i) \tag{18}$$

where *n* is the number of associations in the hypothesis and $P\_j(\chi_i)$ is the probability associated with the Mahalanobis distance for the *i*th association contained in the *j*th hypothesis and is the probability that a measurement will fall within the Mahalanobis distance as explained in Section 2.2.5.4.

To illustrate, suppose we have two targets that we are tracking (T1 and T2) and two measurements (M1 and M2) (See Figure 4). Disregarding the null cases, there are two possible hypotheses (*H_1* and *H_2*). The associations for *H_1* are that T1 gets M1 (T1->M1) and therefore T2 gets M2 (T2->M2). The associations for *H_2* are (T1->M2) and therefore (T2->M1). Suppose M1 is closer to T1 than T2 and the associated probabilities are $P\_1(\chi_1) = 0.9$ and $P\_2(\chi_2) = 0.8$. Similarly, suppose M2 is closer to T2 than T1 and the associated probabilities are $P\_1(\chi_2) = 0.7$ and $P\_2(\chi_1) = 0.6$. In this case, *PAE*(*H_1*) = 1 – [0.9*0.7/((0.9*0.7)+ (0.8*0.6))] = 0.4324 and *PAE*(*H_2*) = 1 – [0.8*0.6/((0.9*0.7)+ (0.8*0.6))] = 0.5676. In this simple example *H_1* is more probable than *H_2* as evidenced by the slightly lower *PAE*. Also, since one minus the *PAE* is the probability the hypothesis is correct, if both hypotheses are retained by a MHT, the probability the MHT has the correct hypothesis is unity. An MHT seeks to retain hypotheses with the lowest *PAE* so that the probability the correct hypothesis is retained approaches unity.

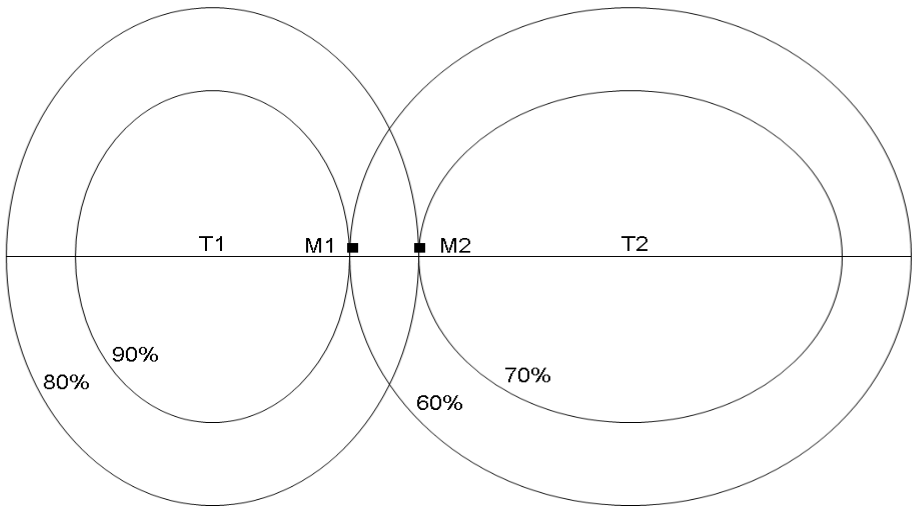


Figure 4 - Illustration of hypothesis scenario. T1 and T2 are the estimated position of two targets. M1 and M2 are two measurements to be associated with T1 and T2. The ellipses represent the curve of equal Mahalanobis distance and the percentages are the associated probabilities.

2.2.5 Ellipses

Since the covariance matrix *P* and innovation covariance matrix *S* can be represented by an ellipse, we need to be familiar with them so we can apply them to tracking. We will cover the ellipse equations, perimeter estimation, and how to obtain the ellipse equation given the innovation.

2.2.5.1 Ellipse Equations

An ellipse centered at (x_0,y_0) with semimajor axis *a* parallel to the *x* axis and semiminor axis *b* (*a*>=*b*) is [35]:

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1 \tag{19}$$

By applying the rotation matrix R_0 [36]:

$$R_0 = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \tag{20}$$

The same ellipse rotated φ degrees is [37]:

$$\frac{((x-x_0)*\cos\varphi+(y-y_0)*\sin\varphi)^2}{a^2}+\frac{((x-x_0)*\sin\varphi-(y-y_0)*\cos\varphi)^2}{b^2}=1 \tag{21}$$

Parametrically, the equation becomes

$$\begin{aligned} x &= x_0 + a\cos(\theta)\cos(\varphi) - b\sin(\theta)\sin(\varphi) \\ y &= y_0 + b\sin(\theta)\cos(\varphi) + a\cos(\theta)\sin(\varphi) \end{aligned} \tag{22}$$

where θ is the varying parameter that varies from 0 to 2π.

2.2.5.2 *Ellipse Perimeter Estimation*

A simple estimate of the ellipse perimeter is [35]:

$$P_e \approx \pi\sqrt{2(a^2 + b^2)} \tag{23}$$

A more accurate estimate is Ramanujan's perimeter estimation [35]:

$$P_e \approx \pi(a+b)\left(\frac{1+3h}{10+\sqrt{4-3h}}\right) \tag{24}$$

where $h = ((a-b) / (a+b))^2$.

2.2.5.3 *Obtaining the Ellipse Equation from Tracking Information*

The ellipse center (x_o,y_o) is the estimated position of the tracked target and is readily available from the state estimate. The innovation covariance of a tracked target gives us the other parameters. The square roots of the eigenvalues of the innovation covariance are the major and minor axis a and b [7]. In addition, φ is determined by finding the rotation matrix R_0 that when multiplied by the innovation covariance makes it diagonal [38]:

$$R_0 S = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix} \tag{25}$$

One can then, solve for φ from the rotation matrix Equation (20). Keep in mind that a zero eigenvalue means that eigenvalues are repeated, hence the ellipse is reduced to a circle with radius a. The center point (x_0,y_0), a, b, and φ are all the parameters needed to fully define an ellipse as per Equation (22).

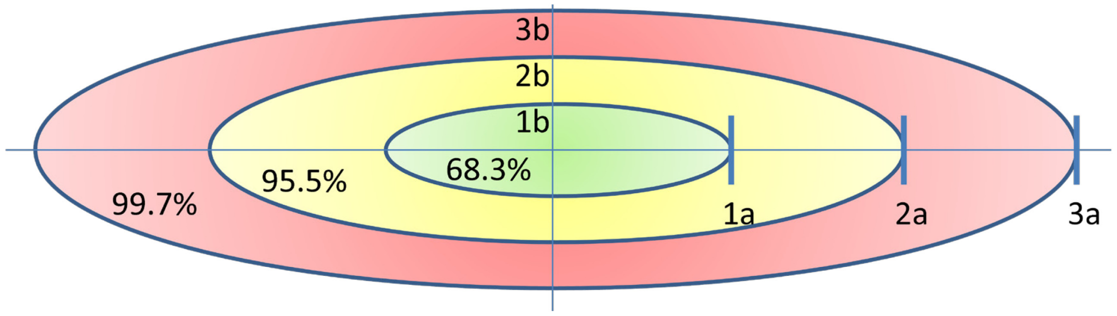


Figure 5 - Relationship between ellipses, Mahalanobis distance, and probabilities. Three concentric ellipses with major and minor axes of *a* and *b* representing a single standard deviation and multiples thereof. The green shaded ellipse has Mahalanobis distance of 1, represents a single standard deviation, and has a 68.3% chance of the measurement falling within the shaded region. The yellow shaded ellipse has Mahalanobis distance of 4, represents two standard deviations, and has a 95.5% chance of the measurement falling within the shaded region (green inclusive). The red shaded ellipse has Mahalanobis distance of 9, represents three standard deviations, and has a 99.7% chance of the measurement falling within any of the shaded regions.

2.2.5.4 Innovation Covariance and Probabilities

Now that we know how to obtain the ellipse values *a* and *b* from the innovation covariance, it is useful to recognize their statistical meaning. The curve of equal Mahalanobis distance is an ellipse. For the problem under consideration, the measurement space is a multi-normal distributed variable with zero means and standard deviations of *a* and *b*. Because of this fact, the Mahalanobis distance measure in Equation (12) results in χ_{ij} being a chi-squared variable whose related probability is used to determine the probability that the measurement belongs to the tracked target and is used in association routines. Therefore, the ellipse created by multiplying *a* and *b* found from Equation (25) by about 3.1 represents 99.99% certainty that any measurement within this ellipse is a valid association for the target under track (note that 3 standard deviations correlates to 99.73%). Throughout this document Pr(χ_{ij}) represents the probability of being within the Mahalanobis distance.

2.2.6 Optimization Problem

An optimal solution is the best solution based on some criteria. The optimization process or lifecycle used to find algorithms to solve an optimization problem has many steps, a few of which are the finding of heuristics, the use of tuning parameters, adaptive tuning parameters, and the use of advanced methods such as evolutionary algorithms.

2.2.6.1 Heuristic Decision Makers

A heuristic decision maker selects between choices using information currently and readily available [39]. A good heuristic correlates well with the optimal solution. The nearest neighbor association algorithm, for example, associates a measurement with the nearest track. These types of algorithms generally produce good solutions but may not produce the optimal solution.

2.2.6.2 Tuning Parameters

A parameter in an algorithm is a number or variable used either in a calculation or as a threshold value. These parameters have come to be called tuning parameters when performance of the algorithm varies as the parameter varies. The measurement and dynamics noise terms, R and Q in a Kalman filter are examples of pertinent tuning parameters [23]. It is well known that optimal performance of the filter depends on the proper values for R and Q. The motion segmentation algorithm provides an example of a threshold tuning parameter. Change is detected if the change in intensity is greater than a threshold value. A small threshold value increases valid detections and clutter while a larger threshold value reduces valid detections and clutter. A proper threshold value will maximize valid detections and minimize clutter.

2.2.6.3 Adaptive Tuning Parameters

An adaptive tuning parameter is generally an improvement over constant valued tuning parameters. The adaptive tuning parameter changes value over time or space. One means of achieving this is by creating a tuning parameter that is a function of time. Suppose for example that clutter at initialization is less tolerable than after tracks are established. In the motion segmentation algorithm we could use an alpha filter $(Re^{-\alpha(t-\tau)})$ where the threshold parameter R slowly increases over the time difference $(t - \tau)$ and the rate is controlled by α. The obvious goal being to achieve better results than those provided with a static threshold. The adaptive Kalman filter illustrates the other and more interesting means of achieving an adaptive tuning parameter [23]. In the update cycle, an estimate for Q is generated by calculating the value of Q that would have no residual. This estimate for Q is then used to alter the current value of Q. This type of adaptive tuning parameter requires either an estimate of the parameter throughout the algorithm, or an estimate of performance (thus an indirect estimate of the parameter) so that the value can be updated.

2.2.6.4 Evolutionary Algorithm

An evolutionary algorithm (EA) is any of a class of algorithms inspired by evolutionary theory [40], [41]. Genetic algorithms (GA) were the first created and share similarities with all evolutionary algorithms. GAs are iterative processes whose flowpath is below.

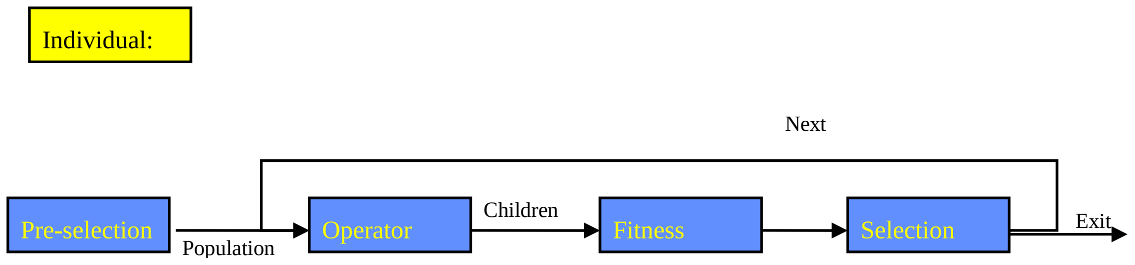


Figure 6 - A typical flowpath of a genetic algorithm.

The *individual* is a potential solution, usually a bitwise representation (i.e., 32-bit number). The pre-selection method generates a list of *individuals* known as the *population.* The initial *population* can be pre-selected randomly, heuristically, or by some other method (such as using another EA). The *population* undergoes operations to generate a new *population* known as *children.* Crossover and mutation are two operations that give the GA its name. In crossover, the bits of two *individuals* (a.k.a. parents) are distributed (while maintaining their relative location) between two new *individuals* (a.k.a. children). This is analogous to the way genetic material is distributed from parents to children. If a bit is thought of as a gene, each child receives genes from the parents, yet each child is different from either parent. Mutation just flips the bit. If a bit is selected for mutation, it is changed to its opposite. The fitness function evaluates the goodness of each *individual*. The selection method provides another analogy with genetics; survival of the fittest. The best *individuals* will be allowed to continue to the next *generation* (iteration) to produce children. The exit criterion (perhaps a set number of iterations, or a time without improvement) stops the algorithm. The best *individual* found throughout the program is reported as the solution. Table 1 provides a summary to help determine what type of optimization method to use:

Table 1- Requirements for implementing optimization features

Algorithm Feature	**What is Needed**	**Real-Time**
Tuning Parameter	Threshold or Equation Values	Y
Adaptive Tuning Parameter	Threshold or Equation Values. Feedback each iteration on how "good" the parameter was.	Y
Heuristics	A priori guideline or rule-of-thumb on how good a selection will be.	Y
Evolutionary	Evaluation equation	N

2.2.6.4.1 Multi-objective Evolutionary Algorithm – NSGA2

The two most predominant multi-objective evolutionary "optimization" algorithms (MOEAs) are NSGA-2 and SPEA-II [41]. NSGA-2 has been shown to produce a slightly superior Pareto front (PF) than SPEA-II for the same population size and number of generations; however the difference is statistically questionable due to overlapping variances. NSGA-2 also converges faster and requires less computational power than SPEA-II for two-objective problems. On the other hand SPEA-II produces a slightly more uniform Pareto front (again statistically questionable). We feel the slightly superior Pareto front of NSGA-2 outweighs the slightly more uniform Pareto front because quality of solution is later measured whereas distribution of solution is not. Additionally, it is the opinion of the author that NSGA-2 code is easier to understand and integrate into the complex tracking fitness function. Based upon these considerations, the NSGA-2 is employed for this study [42].

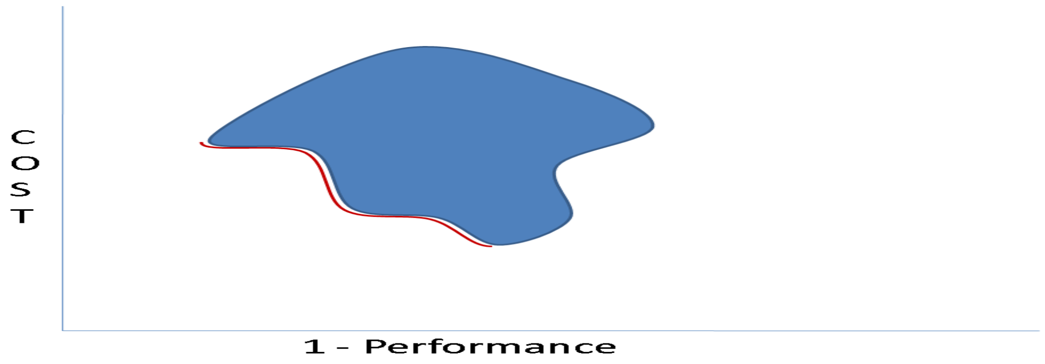


Figure 7 - Illustration of a Pareto front using cost and performance as the objectives. The red curve is the Pareto front for the tradespace of cost vs. performance.

2.2.6.4.2 Pareto Front

The Pareto front is the optimal subset of the tradespace of the objectives [43]. Figure 7 shows a Pareto front with a tradespace of cost vs. performance. In a multi-objective problem formulation, the objectives are always formulated so as to be minimized. Since performance is an objective to be maximized, the Pareto front uses 1 - performance. The tradespace of cost vs. 1- performance is the blue shape in Figure 7 and is not a function since two options may have the same performance, yet cost different amounts. The Pareto front is a subset of the tradespace and is an always decreasing function. Every point on the Pareto front is an optimal solution. The red curve under the blue tradespace in Figure 7 represents the Pareto front and is what most would consider a cost vs. negative performance curve. Although this example shows a continuous curve, a Pareto front may be discontinuous.

2.2.7 Hyperspectral Imaging

An imaging spectrometer makes spectral measurements of bands as narrow as 0.01 micrometers over a wide wavelength range. Typical wavelength range is 0.4 to 2.5 micrometers with a spectral resolution typically 10 mm. Figure 8 illustrates how these images are used to create reflectance vs. wavelength plots for a given pixel [44].

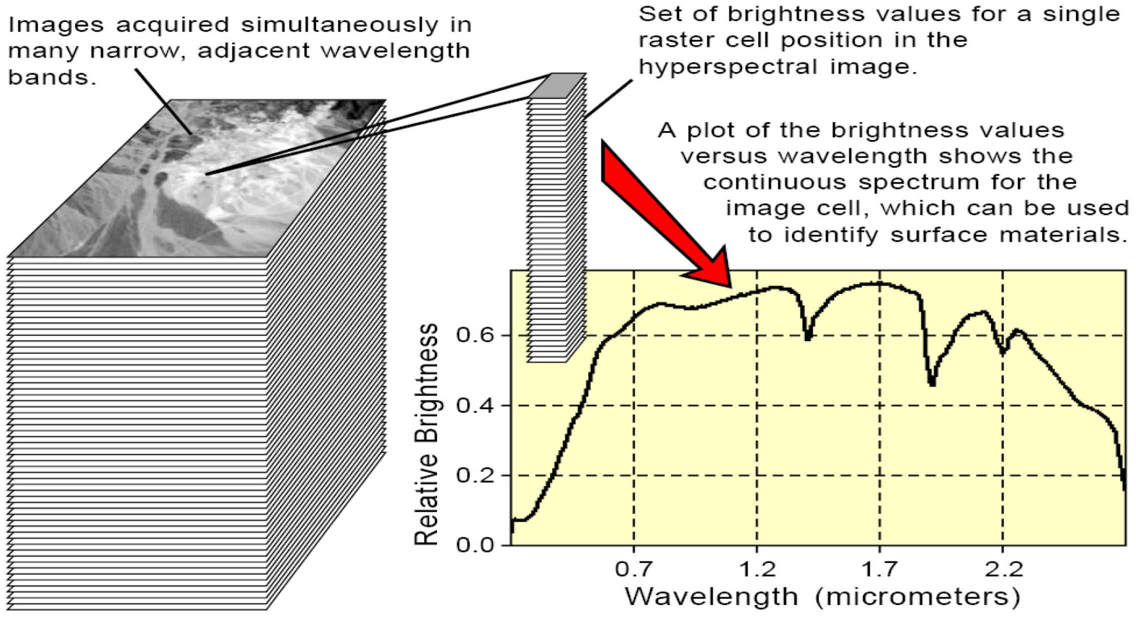

Images acquired simultaneously in many narrow, adjacent wavelength bands.
Set of brightness values for a single raster cell position in the hyperspectral image.
A plot of the brightness values versus wavelength shows the continuous spectrum for the image cell, which can be used to identify surface materials.
Relative Brightness
0.0
0.2
0.4
0.6
0.7
1.2
1.7
2.2
Wavelength (micrometers)

Figure 8 - Illustration of how a reflectance vs. wavelength plot is obtained from a hyperspectral image (image obtained from [44]).

By using the HSI data, we are able to distinguish individual vehicles. Figure 10 shows a plot for 8 black vehicles (perhaps the most challenging color to distinguish). Notice how each plot varies and some are vastly distinct [45].

2.2.8 Hyperspectral Imaging Classifiers

A classifier is an algorithm that finds the closest match of input data compared to a number of preexisting classes. For example, if we take a measurement of one of the 8 black vehicles represented in Figure 9, a classifier will tell us which of the 8 vehicles it is.

There are many types of classifiers and many approaches to deciding which class is correct, each with varying results and tradeoffs. The primary differences between classifiers are what metric is used to compare input data to preexisting classes in order to measure the closeness between them, and the method of optimally arriving at a classification solution.

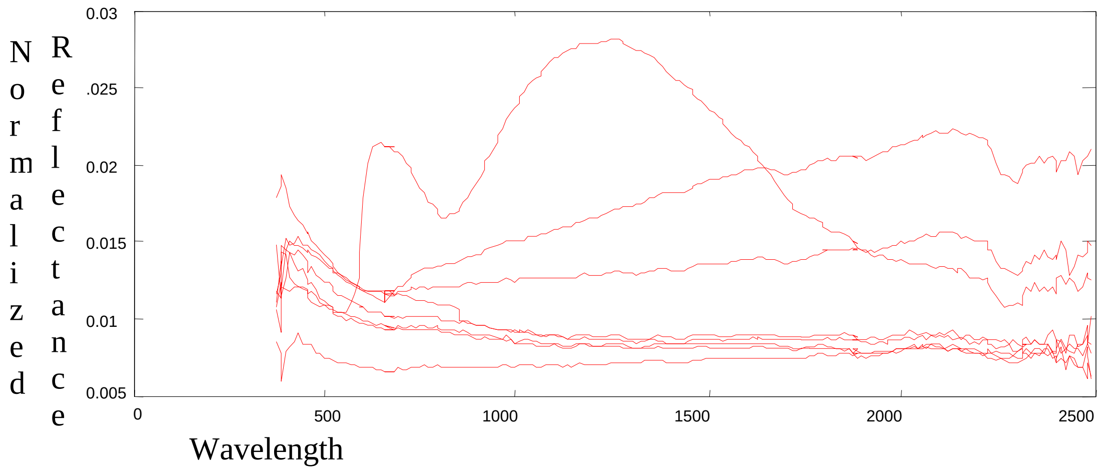


Figure 9 - Reflectance vs. wavelength for eight black vehicles.

The most commonly used measures are distance based measures and orthogonal projection based measures. Three ways to calculate the distance between the spectral signatures of two pixel vectors, s_i and s_j and can be derived from the l_1, l_2, and l_∞ –norms are:

City block distance (CBD) corresponding to the l_1-norm

$$CBD(s_i, s_j) = \sum_{l=1}^{L} | s_{il} - s_{jl} | \tag{26}$$

Euclidean distance (ED) corresponding to the l_2-norm

$$ED(s_i, s_j) = \| s_i - s_j \| \equiv \sqrt{\sum_{l=1}^{L} (s_{il} - s_{jl})^2} \tag{27}$$

Tchebyshev distance (ED) or maximum distance corresponding to the l_∞ -norm

$$TD(s_i, s_j) = \max_{1 \le l \le L} \{| s_{il} - s_{jl} |\} \tag{28}$$

Two orthogonal projection measures are spectral angle mapper (SAM) and orthogonal projection divergence (OPD)

$$SAM(s_i, s_j) \equiv \cos^{-1}\left(\frac{s_i \cdot s_j}{\| s_i \| \ \| s_j \|} \right) = \cos^{-1}\left(\frac{\sum_{l=1}^{L} s_{il} s_{jl}}{\sqrt{\sum_{l=1}^{L} s_{il}{}^2} \sqrt{\sum_{l=1}^{L} s_{jl}{}^2}} \right) \tag{29}$$

$$OPD(s_i, s_j) = \sqrt{\left(s_i{}^T P_{s_j}^{\perp} s_i + s_j{}^T P_{s_i}^{\perp} s_j \right)} \tag{30}$$

where the projection $P_{s_k}^{\perp} = \mathrm{I}_{LxL} - s_k (s_k^T s_k)^{-1} s_k^T$ for $k = i,j$ and $\mathrm{I}_{\mathrm{LxL}}$ is the identity matrix

Additional metrics beyond distance or orthogonal projection include spectral information divergence (SID), hidden Markov model-based information divergence (HMMID), and others [46].

The method of optimally reaching a classification solution is performed by a variety of optimization techniques. These methods perform a combination of feature subset selection and data priority ordering. In feature subset selection only a portion of the HSI data is used, thus requiring fewer calculations. The selected subset of data is prioritized to take advantage of the fact that some classes require less data to classify than others. There are parametric classifiers based on multivariate statistical models such as the Gaussian maximum likelihood method (ML) [47]. Evolutionary algorithms are also used in classifiers [21]. Neural networks are the most prevalent optimization technique used in the literature with generalized learning vector quantization (GLVQ) and generalized relevance learning vector quantization (GRLVQ) as examples [48].

When comparing classifiers, the most commonly used method is classification accuracy. Many works use classification accuracy exclusively without regard to computational performance since the application is not real-time [48], [49], [50]. Since a target tracker is a real-time application, the computational performance of the classifier is vital to the successful real-time implementation and likewise needs to be measured [51]. Since a large number of classifiers rely on neural networks, some neural network performance measures are also used, such as classification accuracy vs. number of training samples [52]. There are also external considerations that need to be thought through. Since we will be adding and removing classes on-the-fly, how will the algorithm maintain performance and what will be the computational burden? The selection/creation of the best classifier for use with the utility function is a highly complex multi-objective problem beyond the scope of this dissertation. The experimentation in this work does not use a classifier or hyperspectral imagery, but rather characterizes the performance of a classifier with the probability of correct classification (PCC) which is a reflection of the classification accuracy. Furthermore, the PCC as used in this work is a Gaussian variable, however there is no assurance that classification accuracy is in fact Gaussian. It should further be noted that in a hyperspectral target detection and classification algorithm, high target detection rates do not necessarily result in high classification accuracy rates [53]. In any case, several values for PCC will be used to represent variations in classifier performance in order to support overall system performance.

If optimization is not used, a classifier would compare every input pixel vector with all other pre-existing classes according to the selected measure and choose the class that has the minimum compared to the input vector. This exhaustive method would be highly computationally expensive. The total number of evaluations per frame of data would be # pixels x # HSI bands x # pre-existing classes which for a single frame of data in this work is about 2048 x 1536 x 200 x 100 = 6.29 x 10^{10}. The computational burden is compounded by the fact that the various calculations (such as Equation (29)) are computationally complex and the number of frames over time is large. Spectral feature selection reduces the number of bands required in the calculation. The beauty of spatial sampling as presented in this work is that it seeks to reduce the number of pixels that need to be classified (which is the largest factor), thus the computational burden for the classification task is drastically reduced, yet all classifier optimization work is still leveraged for the pixels that are evaluated.

2.3 Chapter Summary

With the related works covering the intersection of HSI, RSM, optimization and target tracking, and the background information of models and states, filtering, target tracking, the probability of association error, ellipses, optimization, and HSI that were presented in this chapter, we are ready to formally define the problems and present potential solutions in the next chapter.

3 Problem Formulations and Proposed Solutions

The primary problem being solved is spatial sampling of HSI data for target tracking. Part of this solution is determining the predictive probability of association error (PPAE). Prior to discussing PPAE, there are several related problems to be solved that aid in understanding the solution to PPAE. These problems are determining the probability a target is within an arbitrary region and determining the joint probability that two targets overlap and are within the same arbitrary region.

Spatial sampling also requires the tracker to be tuned. The Monte Carlo multi-target tracking multi-objective tuning problem is presented along with the tuning method used with the tracker prior to spatial sampling. Finally, the spatial sampling problem is presented which includes PPAE.

3.1 Determining the Probability a Target is within an Arbitrary Region

The problem of interest is given an ellipse representing a target being tracked in two dimensions: find Pr(*A*), the probability the target is within an arbitrary region *A*.

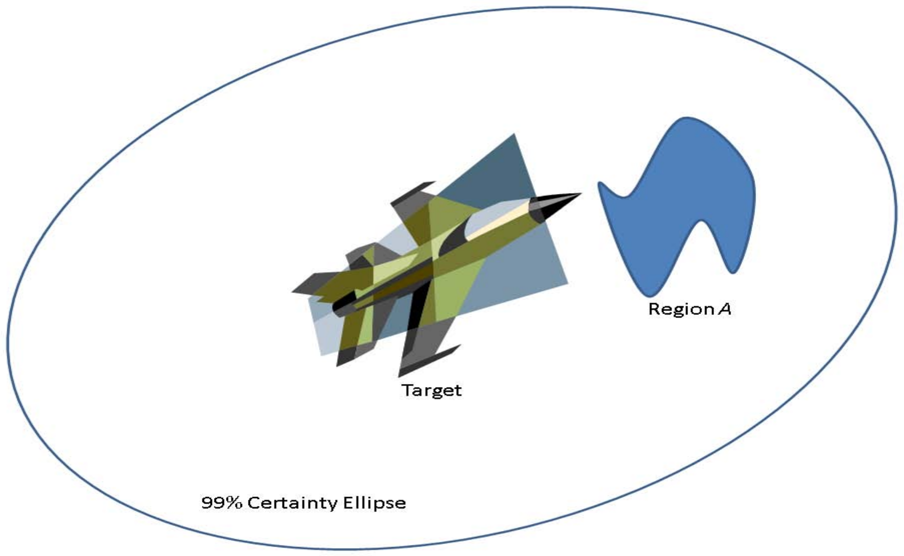


Figure 10 - A target position estimate and an arbitrary region A are surrounded by an ellipse representing 99% certainty of target position.

In Figure 10, the target estimated position is the center of the ellipse. The target has a 99% certainty of being within the ellipse based on the kinematic tracking of the target as discussed in Section 2.2.5.4. Although the arbitrary region *A* depicted in this example is contained wholly in the ellipse, it can cross the boundary, however any area outside the ellipse can be ignored since the probability of the target position being anywhere outside the ellipse is small (in this case 1%). The probability the target is within an arbitrary region Pr*(A)* is the integral of the probabilities of every point within the region *A*.

$$\Pr(A) = \int_A \Pr(x, y)\,d(x, y) \tag{31}$$

A natural way to view this is to think of the point (*x*,*y*) as a small square *dxdy*. This leads to

$$\Pr(A) = \iint_A \Pr(x)\Pr(y)\,dxdy \tag{32}$$

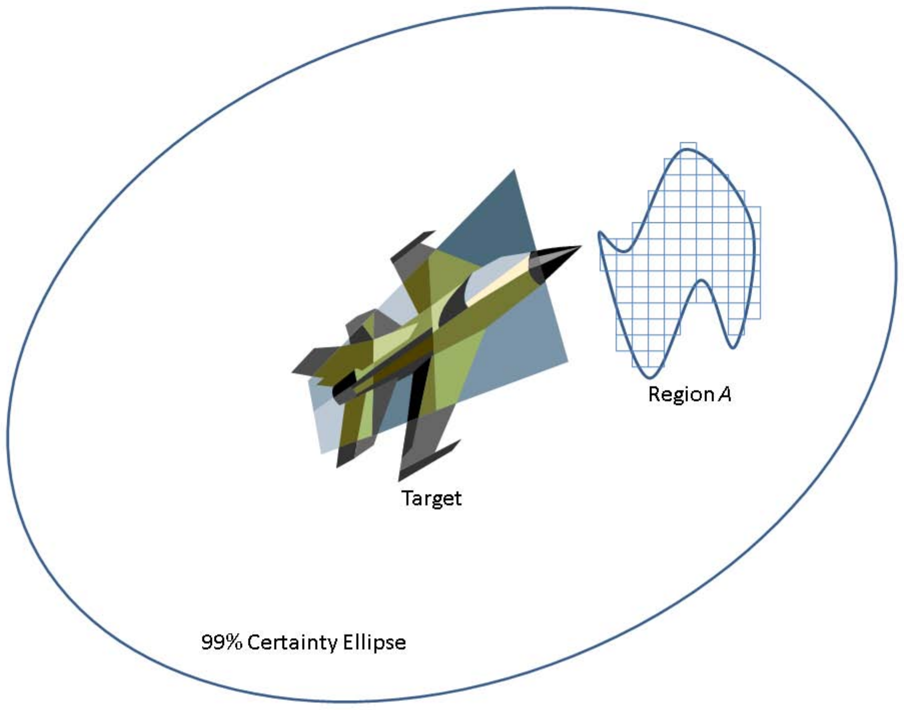


Figure 11 - Approximating arbitrary region probability by small squares.

Figure 11 shows the arbitrary region *A* approximated with many small squares.

Since we are working with a conic section, another natural way to view this is in terms of small arcs; which leads to

$$\Pr(A) = \iint_A \Pr(r)\Pr(\phi)\,dr\,d\phi \tag{33}$$

Figure 12 shows the arbitrary region *A* approximated by small arcs. See also Figure 13 and Figure 25 for a better understanding of small arcs.

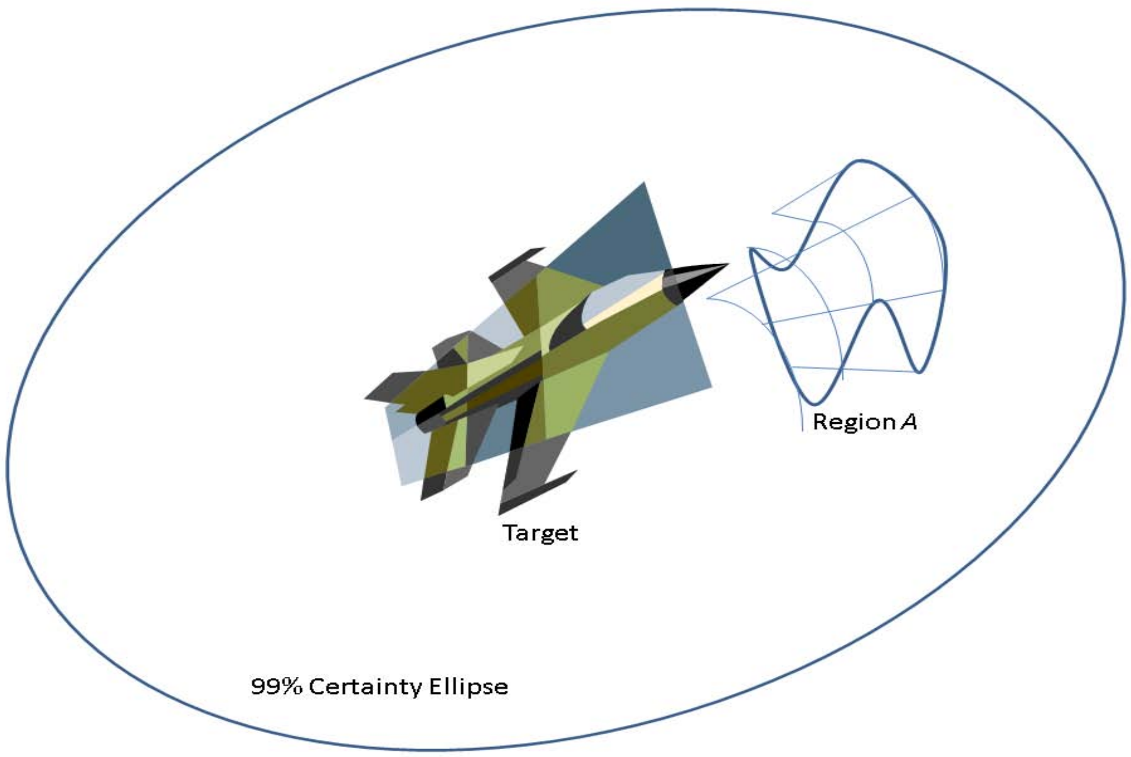


Figure 12 - Approximating arbitrary region probability by small arcs.

3.1.1 Numerically Solving by Small Squares

To numerically approximate the solution to Equation (32), we change the integral to a summation and find Pr(*x*)*dx* and Pr(*y*)*dy* through a difference in the cumulative density function (C*DF*)

$$\Pr(A) \approx \sum_{A} (CDF(x+.5dx) - CDF(x-.5dx)) * ((CDF(y+.5dy) - CDF(y-.5dy)) \qquad (34)$$

where the ellipse and the region *A* are first rotated so that the semimajor axis *a* is parallel to the x-axis, *a* and *b* refer to the semimajor axis and semiminor axis of the ellipse, and are equivalent to one standard deviation as used in the *CDF* formula for *x* and *y* respectively. The difference in the *CDF* function is the probability of being within the endpoints [54]:

$$CDF(x+.5dx)-CDF(x-.5dx)=\Pr(z: x-.5dx \le z \le x+.5dx) \quad (35)$$

While it may seem intuitive to have a *dxdy* term somewhere in Equation (34) to account for the area, the difference in *CDF*s already accounts for this. Consider that as *dx* decreases, the probability of being within the endpoints also decreases, hence the *dx* term is inherently in the difference in *CDF*s.

3.1.2 Numerically Solving by Small Arcs

The probability of the target being in an elliptical band centered at *r* is a difference in probabilities similar to Equation (35):

$$\Pr(r+.5dr)-\Pr(r-.5dr)=\Pr((x,y):\ r-.5dr \le \chi^2 \le r+.5dr) \quad (36)$$

where χ^2 is the chi-squared distance of (*x*,*y*) to the center of the ellipse.

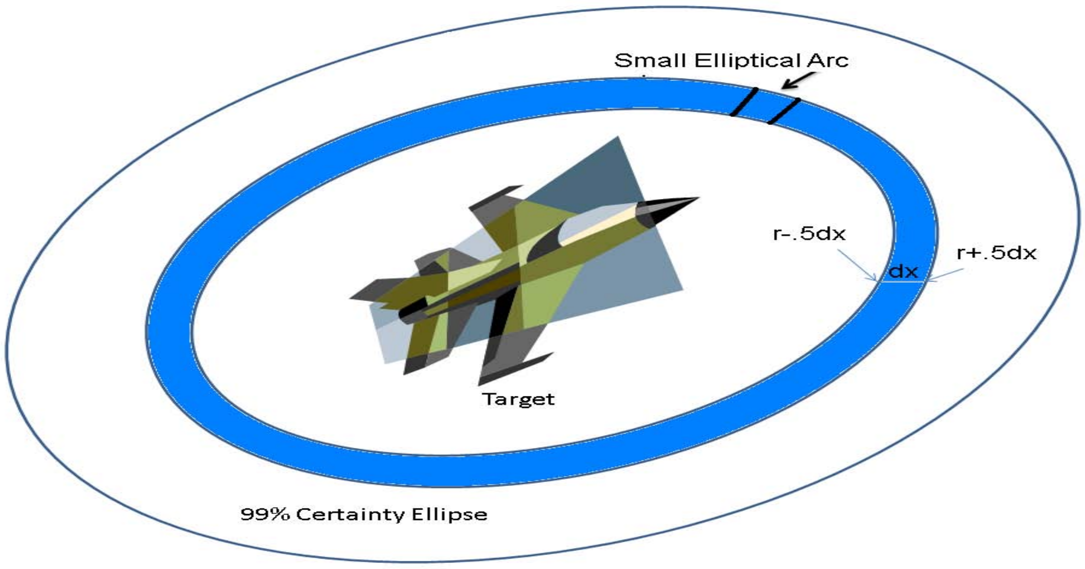


Figure 13 - Depiction of an elliptical band and small elliptical arc.

Figure 13 shows a blue elliptical band centered a distance of *r* from the center with a thickness of *dr*. A portion of this band is a small elliptical arc.

In order to obtain a small arc, we need only take a portion of the elliptical band. In the case of a circle, this led to the observation that every angle has equal probability and that $\Pr(\phi)d\phi = d\phi/2\pi = rd\phi/2\pi r$. The second equality says to find the arclength and divide by the circle circumference. For an ellipse, we initially attempted to likewise take the arclength and divide by the Rumanujan elliptical circumference Equation (24); however we soon realized equal arclengths do not have equal probability. This led to using the parametric form of the elliptical Equation (22) substituting ϕ for θ to determine values in the (*x*,*y*) space. The numeric solution is

$$\Pr(A) \approx \sum_{A} (\Pr(r+.5dr) - \Pr(r-.5dr)) * (d\phi/2\pi) \tag{37}$$

3.1.3 Boundary Overlap Error

Boundary overlap error is caused by trying to approximate the arbitrary boundary by either small squares or small arcs. For example, a triangle cannot be precisely modeled by either small squares or small arcs. Some boundary portions of the triangle will either not be covered, or will have additional area over the boundary. This error factor is minimized as the squares or arcs become smaller.

3.2 Determining the Joint Probability Two Targets Overlap

The only way an association error can occur is if the two innovations overlap. While this joint probability is not the same as the PPAE, the numeric calculations are very similar since both use probabilities of the targets over the same region. This joint probability has been used in applications where the PPAE would be more applicable since it is expected they correlate well. Here is the formal problem formulation.

Given two ellipses representing 99% certainty of target location where the probabilities for two targets being tracked are mutually exclusive, determine the probability that both targets are in the region where both ellipses overlap.

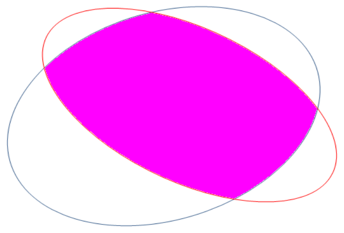


Figure 14 - The magenta region is the overlapping area of two ellipses.

Since the probabilities for the two targets are assumed to be independent, the joint probability both targets overlap is simply the product of the probabilities the targets are within the region of overlap. These probabilities are numerically determined and approximated as per Section 3.1.

$$\Pr_{12}(A) = \Pr_{1}(A) * \Pr_{2}(A) \tag{38}$$

The joint probability of target overlap $\Pr_{12}(A)$ is later compared to PPAE as used in spatial sampling.

3.3 Spatial Sampling

The goal of spatial sampling is to determine which pixels will collect HSI data. A utility function is formulated as a linear combination of heuristic values that assigns a value to the usefulness of collecting HSI data at each pixel. This utility function is mission dependent and may even be data dependent. We assume an urban target tracking mission and a kinematic tracker using the electro-optical imagery data. For every target being tracked, the following information is available from the kinematic tracker:

$\hat{X}$: The estimate of the position and velocity in pixels.

P: The uncertainty of $\hat{X}$.

R: The measurement uncertainty.

H: An output matrix relating measurements to the state vector

S: The innovation ($S = HPH' + R$).

r_{ij}: The residual or difference between the *ij*th pixel and $\hat{X}$.

χ_{ij}: The Mahalanobis distance measure from Equation (12) between the *ij*th pixel and $\hat{X}$. This distance measure is a realization of a chi-squared variable[26].

$\Pr(\chi_{ij})$: The probability that a measurement falls within the specified Mahalanobis distance. See Section 2.2.5.4 for a discussion on how this probability is obtained. Note that as the measurement becomes closer to the estimate, the Mahalanobis distance decreases and this probability decreases. Thus, 1 - $\Pr(\chi_{ij})$ is a measure of how likely a specific measurement came from the track.

3.4 Utility Function Components for Spatial Sampling

Let $U_{ij}(t)$ represent the utility of obtaining HSI data at the ij^{th} pixel at time t. The value of $U_{ij}(t)$ depends on various factors such as the probability associated with the Mahalanobis distance from a target of interest (TOI) to the ij^{th} pixel which can be expressed as $\Pr(\chi_{ij})$ and is discussed in Section 2.2.5.4. Additional contributors to utility function values are:

$U_{ij}^{D}(t)$: Default value that every TOI receives

$$U_{ij}^{D}(t) = 1 - \Pr(\chi_{ij}) \tag{39}$$

Thus, the utility of gathering HSI data at $\hat{X}$ is 1 and gradually decreases toward 0 as we get farther from $\hat{X}$. This and all further utility function components contain the term $1 - \Pr(\chi_{ij})$ to represent the effect of the kinematic distance from the TOI. Since $1 - \Pr(\chi_{ij})$

uses the innovation covariance from the TOI, the values of this default utility component follow the same bi-normal distribution about the target estimate.

$U_{ij}^{N}(t)$: **N**ew model utility which is a function of the appearance of new or reacquired targets that need to be sampled in order to build a target feature model.

$$U_{ij}^{N}(t) = \begin{cases} 1 - \Pr(\chi_{ij}) & \text{if the target has not been HSI sampled} \\ 0 \text{ otherwise} & \end{cases} \tag{40}$$

$U_{ij}^{A}(t)$: Association utility defined for closely spaced TOIs where track state and the related uncertainty provide a measure of association ambiguity.

$$U_{ij}^{A}(t) = 2 * PPAE * (1 - \Pr(\chi_{ij})) \tag{41}$$

Where the predictive probability of association error (PPAE) is a value from 0 to .5 (thus the 2* makes it from 0 to 1) and is the probability the TOI will cause an association error based on its nearness to other TOIs and their track states. PPAE is further discussed below in Section 3.5 where the problem formulation and numeric solution are presented.

$U_{ij}^{M}(t)$: Missed measurement utility which is a function of the number of missed detections (*m*) for the kinematic tracker due to occlusion or shadow. This value can also greatly be aided by scene context. If the missed measurements are due to a building, there is no sense in gathering HSI data, however if a vehicle enters a shadowed region, HSI data may provide a measurement even though electro-optic video fails to do so.

$$U_{ij}^{M}(t) = (1 - e^{-m}) * (1 - \Pr(\chi_{ij})) \tag{42}$$

where *m* is the number of missed measurements in the last *n* measurements.

$U_{ij}^{\Im}(t)$: model age which is a function of the time since the last spectral model measurement was incorporated

$$U_{ij}^{\Im}(t) = 1 - e^{-\alpha(t-\tau)} * (1 - \Pr(\chi_{ij})) \quad (43)$$

where τ is the time of the last HSI sample of a pixel in the TOI and α is a decay rate value to control the growth of the utility function.

The aggregate of the above terms gives the total utility function:

$$\begin{aligned} &U_{ij}(t) = C^{D}U_{ij}^{D}(t) + C^{N}U_{ij}^{N}(t) + C^{A}U_{ij}^{A}(t) + C^{M}U_{ij}^{M}(t) + C^{\Im}U_{ij}^{\Im}(t) \\ &\text{s.t. } U_{ij}^{\Phi}(t) \in [0,1], \quad \Phi \in \{D, N, A, M, \Im\}, \quad \sum C^{\Phi} = 1, \quad C^{\Phi} \geq 0 \ \forall \ \Phi \end{aligned} \quad (44)$$

The values of *C* are the relative importance or weighting of the different utility components. It should be obvious that these values tune the utility function. What may not be so obvious is that the selection of TOIs also tunes the utility function. This enables some inherent robustness. For ease of implementation, the new model utility is combined with the model age utility. When a track is initiated, the initial time is set so as to maximize the model age utility, making it equal to the new model utility.

3.5 Predictive Probability of Association Error (PPAE) Problem Development

3.5.1 Difference between Probability of Association Error and PPAE

The probability of association error (PAE) was presented in Section 2.2.4. How is PAE different from the predictive probability of association error (PPAE)? Probability of association error is reactive. Given the measurements and the track association solution arrived at by the tracker, it seeks to answer the questions "How probable is it that the association is correct?", and "If the association is in error, where did it go wrong?" This information is used to generate alternate hypothesis for a multi-hypothesis tracker (MHT) to maximize the total probability of correct association for the resources available.

In contrast, PPAE introduced in this dissertation is proactive. Prior to receiving the measurements, it seeks to answer the questions "How probable is it that an association error will occur?", and "How probable is it that a specific target will have an association error?" This information is used in a control feedback loop to determine which targets require additional information, or in the case of a multi-modal sensor, which areas require the higher mode of operation. PPAE seeks to resolve potential association errors through the gathering of additional information. After the measurements are obtained, PAE can still be used in conjunction with a MHT. The additional information obtained by using PPAE when combined with using PAE should result in fewer required hypothesis and greater total probability of correct association for the resources available.

3.5.2 1-D Problem Development of PPAE

We know the probability distribution function (PDF) of each target in terms of kinematic uncertainty, and based on the Kalman filtering approach used in the tracker, we can calculate the PDF of each measurement. We then determine the track-to-measurement association and compute the predictive probability of association error.

3.5.2.1 Formulation of the 1-D PPAE Problem

Given the Gaussian PDF of two tracks, $T_1 \sim N(0, \sigma^2_1)$ and $T_2 \sim N(\mu_2, \sigma^2_2)$ with mean and variances as given, and given two measurements Z_1 and Z_2 corresponding to T_1 and T_2 with measurement uncertainty $R \sim N(0, \sigma^2_R)$, find the predictive probability of association error while minimizing the total distance from target estimates to measurements as a method of association.

It is assumed that the given PDFs for the track state of T_1 and T_2 are accurate Gaussian representations. In a tracker, if association errors or track swaps have occurred, or if the dynamic or uncertainty model is not correct, or if the tracker is not properly tuned, the resulting covariance matrix *P* from Equation (10) is a misrepresentation of the true PDF. We therefore assume these errors have not occurred and that *P* may be used to satisfy the given PDFs.

The goal of an association method is to minimize the total distance from measurements to estimates. This minimizes the probability of association error. With only two targets and measurements, it is not challenging to find the minimum distance. Other methods exist because finding the minimum distance is an NP-complete problem and as the number of targets and measurements increases the computational power required to solve for the minimum distance grows geometrically becoming computationally infeasible. This optimal approach will provide a useable estimate for other association methods such as nearest neighbor, 2-cut, or auctioning presented in Sections 2.2.3.2.3 and 2.2.3.2.4. These other association methods are non-optimal, but near-optimal. Hence, the PPAE found using this assumed optimal association method will be slightly lower than the true PPAE using one of the near-optimal association methods.

In the above formulation, the mean of T_1 is equal to zero in order to simplify the problem. It can, in general, take on any value. Since the PPAE does not depend on the mean of T_1 but depends on the distance between the means of the targets, choosing a mean of zero makes sense.

3.5.2.2 Measurement PDF

Since we have the PDF of each target in terms of kinematic position, we can calculate the total PDF of a measurement resulting from each target. This is simply $Z = T + R$. Thus $Z_1 \sim N(0, (\sigma^2_1 + \sigma^2_R)) = N(0, \sigma^2_1) + N(0, \sigma^2_R)$ and likewise $Z_2 \sim N(\mu_2 + 0, (\sigma^2_2 + \sigma^2_R)) = N(\mu_2, \sigma^2_2) + N(0, \sigma^2_R)$. We then apply the association rule and determine the predictive probability of association error.

3.5.2.3 Association Rule

The 1-D development allows for a simplified association rule. In Figure 15, the two measurements PDFs Z_1 and Z_2 are depicted along with a specific realization of Z_1 and Z_2. In order for an association error to occur, the Euclidean distance from the mean of T_1 to the realization Z_1 plus the Euclidean distance from the mean of T_2 to the realization Z_2 must be greater than the Euclidean distance from the mean of T_1 to the realization Z_2 plus the Euclidean distance from the mean of T_2 to the realization Z_1. This happens when the realization of Z_2 is less than the realization of Z_1.

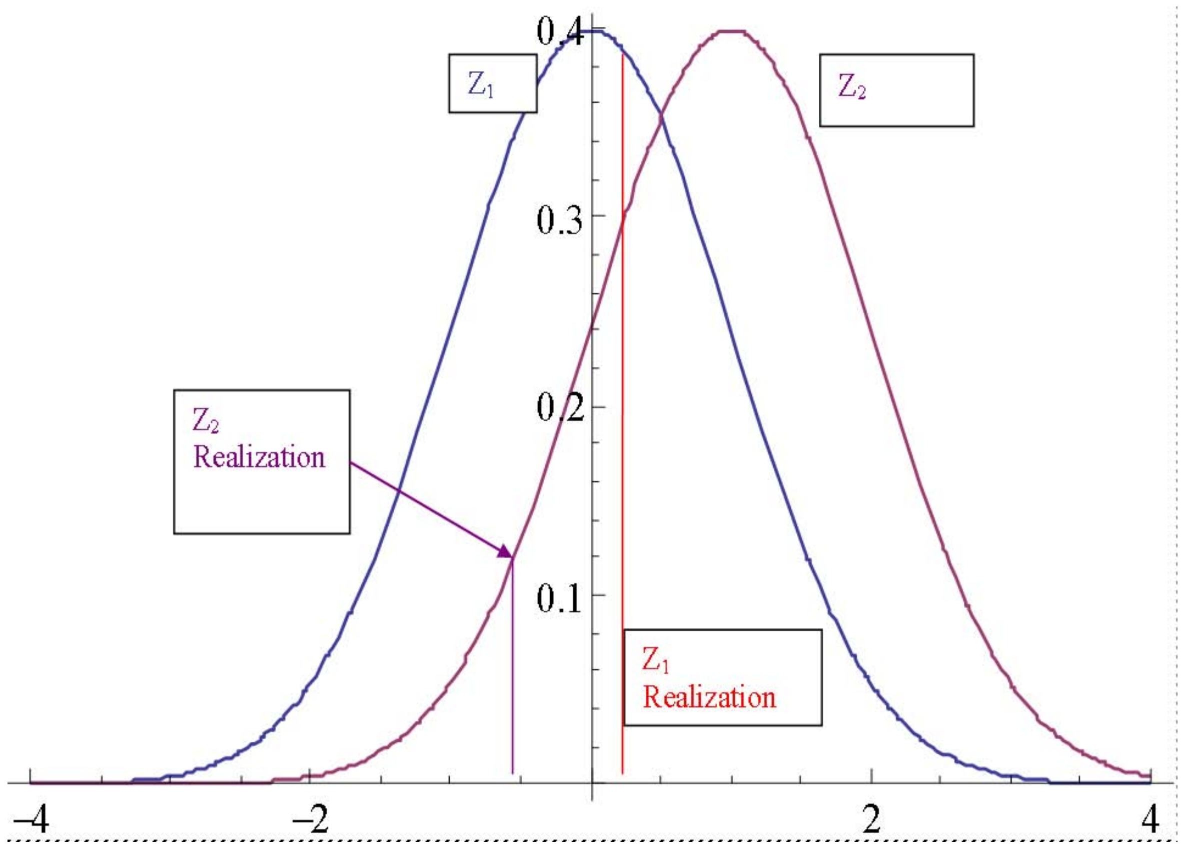


Figure 15 - Illustration of an association error.

3.5.2.4 1-D PPAE

Since we know the probability of a specific realization of Z_1 (PDF(Z_1,x)), we can multiply that by the probability of a specific realization of Z_2 (PDF($Z2$,y) to get the joint probability for both x and y. If we integrate over all realizations of Z1 (dx) and for those values of $Z2$ which cause an association error ($y < x$), we have the PPAE. This is equivalent to taking half the area of the joint PDF.

$$PDF(\mu,\sigma^2,x) = e^{\frac{-(x-\mu)^2}{2\sigma^2}} / (\sqrt{2\pi\sigma^2}) \qquad (45)$$

$$PPAE = \int\int_{-\infty}^{x} PDF(Z_1,x)PDF(Z_2,y)\,dxdy \qquad (46)$$

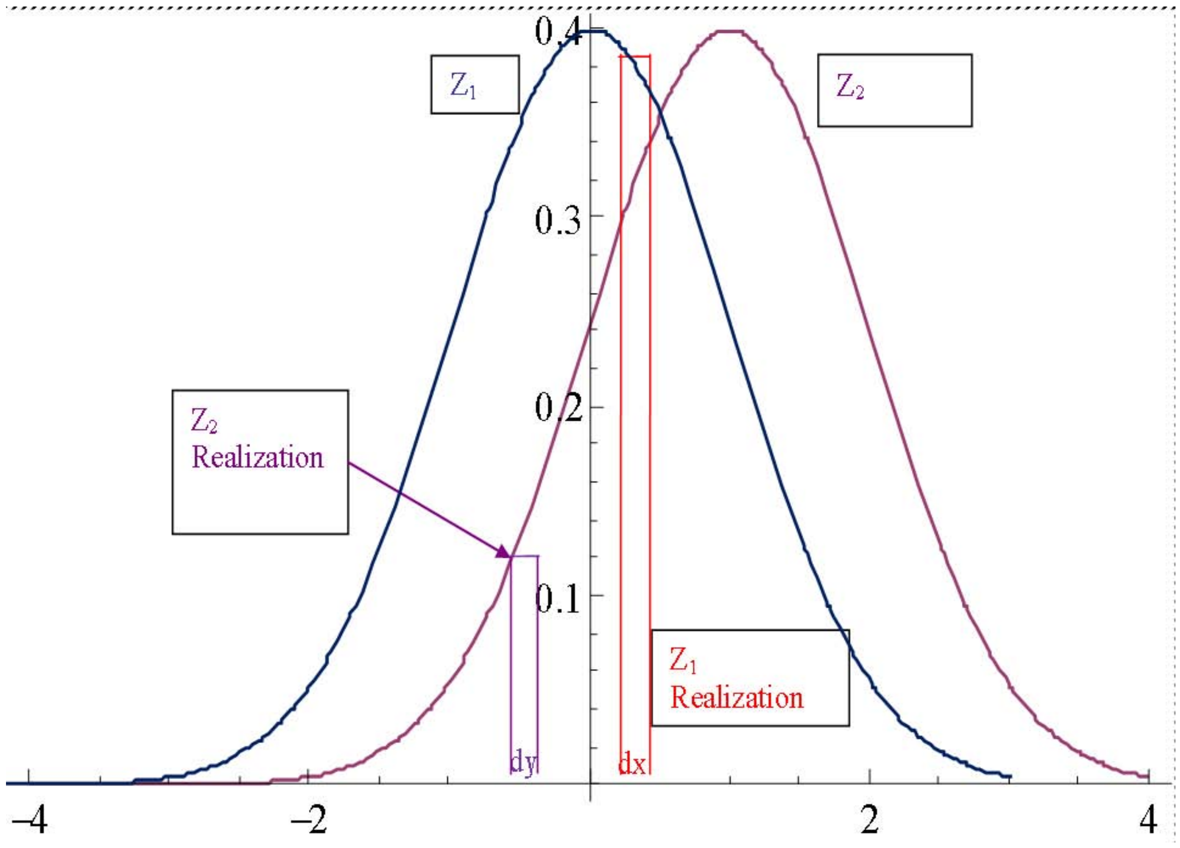


Figure 16 - Illustration of Equation 46. The PDF for every realization Z_1 is multiplied by the PDF for every realization Z_2 where y<x.

The probability of a realization being less than a specific value is captured by the cumulative density function, thus an alternate form of the equation is:

$$PPAE = \int PDF(Z_1, x) CDF(Z_2, x)\, dx \tag{47}$$

3.5.3 2-D Discussion

Now that we have found an integral solution for the 1-D problem, it seems reasonable that the same approach be tried with multiple dimensions. Rather than formally restating the problem as in Section 3.10.1.1, we'll start with the assertion that the problem is similar with the exception of finding the PDF for Z_1 and Z_2, and in the association rule.

3.5.3.1 Measurement PDF

In the 1-D problem, the PDF for the target estimation uncertainty and the measurement uncertainty are both Gaussians. In the 2-D problem, they are both multivariate Gaussian. In the 1-D problem, we solved for the total uncertainty of a measurement of a specific target in Section 3.10.1.2. For the 2-D problem, this PDF is solved in the tracker and is obtained from the innovation covariance matrix *S* as discussed in Sections 2.2.5.3 where we discuss obtaining the ellipse equation from the covariance and innovation covariance and 2.2.5.4 where we discuss the relationship between the innovation covariance and probabilities.

3.5.3.2 Association Rule

In the 1-D formulation, we determine the point at which an association error occurs in Section 3.5.2.3. In other words, we find the point at which association is equal. This point is easy to determine as the value when the realization of Z_1 equals the realization of Z_2. . We then use this value as the limit of integration in the integral Equation (46) to solve for the probability. In the 2-D formulation, we must use kinematic distance and instead of a point where the probability of association is equal for a given realization of Z_1, we have a curve (see Figure 24). This curve is not easily solvable in closed form. In the 1-D case, the limit of integration was the point *x*. In the 2-D case, it will be the curve of equal association. Thus, instead of Equation (46) from the 1-D case, in the 2-D case we get the integral:

$$PPAE = \int \int_{f2(\vec{\xi})}^{f1(\vec{\xi})} PDF(Z_1, \vec{\xi}) PDF(Z_2, \vec{\psi}) d\vec{\xi} d\vec{\psi} \tag{48}$$

where $\vec{\xi}$ is the vector (x_1,y_1) for the realization of Z_1 , $\vec{\psi}$ is the vector (x_2,y_2) for the realization of Z_2 , and $f1(\vec{\xi})$ and $f2(\vec{\xi})$ are used to represent the area of association error for when the realization of Z_1 is the vector $\vec{\xi}$. While we cannot solve for $f1(\vec{\xi})$ and $f2(\vec{\xi})$, if we are given the vectors $\vec{\xi}$ and $\vec{\psi}$, we can determine if they cause an association error which is the purpose of $f1(\vec{\xi})$ and $f2(\vec{\xi})$. This leads to an alternate form:

$$PPAE = \iint A(\vec{\xi},\vec{\psi})PDF(Z_1,\vec{\xi})PDF(Z_2,\vec{\psi})d\vec{\xi}d\vec{\psi} \tag{49}$$

$$\text{Where } A(\vec{\xi},\vec{\psi}) = \left\{ \begin{array}{l} 1 \text{ if } (\vec{\xi},\vec{\psi}) \text{ causes an association error} \\ 1/2 \text{ if } (\vec{\xi},\vec{\psi}) \text{ is indeterminate or equal} \\ 0 \text{ if } (\vec{\xi},\vec{\psi}) \text{ does not cause an association error} \end{array} \right\} \tag{50}$$

A property of A is that $A(\vec{\xi},\vec{\psi}) + A(\vec{\psi},\vec{\xi}) = 1$. If $(\vec{\xi},\vec{\psi})$ causes an association error, $(\vec{\psi},\vec{\xi})$ cannot cause an association error since that merely reverses the pair of measurements and hence reverses the association error. This property enables a nearly twofold speedup in the implementation of the numeric integration since $A(\vec{\xi},\vec{\psi})$ and $A(\vec{\psi},\vec{\xi})$ can be handled simultaneously.

3.5.3.3 Numeric Calculation of PPAE by Summation

Equation (49) can be numerically approximated using summation.

$$PPAE \approx \sum_{\vec{\xi}} \sum_{\vec{\psi}} A(\vec{\xi},\vec{\psi})CDF(Z_1,\vec{\xi})CDF(Z_2,\vec{\psi}) \tag{51}$$

where the CDF functions for the small squares are found by using Equation (35).

3.5.3.4 Errors in the Numeric Calculation of PPAE by Summation

In addition to the error discussed in Section 3.1.3, this numeric calculation introduces an area misalignment error. The overlapping area is approximated using small squares; however the small squares are oriented in relation to the major and minor axis of the ellipse. This is necessary in order to find the difference in CDF for the small square. If the major and minor axes for both ellipses do not share the same reference frame, the approximation of the overlapping area will differ for the two ellipses since the small squares used for one ellipse will be tilted in relation to the other ellipse.

3.6 Optimization of the Utility Function

With the information provided by the PPAE in hand, we desire to solve the following optimization problem: Let N be a subset of the utility values for all pixels in a frame of data $\{U_{i1j1}(t), U_{i2j2}(t), \ldots, U_{i(n-1)j(n-1)}(t), U_{injn}(t)\}$

where U_{injn} is the utility value of the nth element in N and (in,jn) is the pixel location associated with the utility value.

We wish to find the n pixels with the maximal value:

$$\max \sum_{n} U_{injn}(t) \qquad \text{s.t. } U_{injn}(t) \in N \tag{52}$$

Since we know the TOIs based on the current set of tracks, it is fairly straightforward to find which pixels are used in N, perhaps without even calculating their utility. What we do not know are the values of C. These values can be determined with respect to n with a multi-objective optimization problem formulation.

$$\begin{aligned}&Minimize\ F = (f1, f2), where\\&f1 = FMT(scenario)\\&f2 = IDE(scenario)\end{aligned} \tag{53}$$

where *FMT*(*scenario*) is from Equation (14), and *IDE*(*scenario*) is from Equation (15).

In this problem, an *individual* consists of the 5 values of *C* (5 real values between 0 and 1) to be found to satisfy Equation (44). The *population* is a collection (typically of size 100) of *individuals*. A *generation* consists of running the tracking program (using the values of C from each *individual* in the *population*) on a given set of data (consisting of several hundred frames of representative data). The functions *f1* and *f2* will be minimized when the correct HSI data is gathered; hence the associated weightings will be our solution.

As with any control feedback loop, the delay from the control signal to the action on the signal affects the results. We assume an instantaneous feedback (our decisions affect the next frame collection) in the control feedback loop.

3.7 Chapter Summary

We have presented problem formulations and proposed solutions to determining the probability a target is within an arbitrary region, determining the joint probability two targets overlap and are within the same arbitrary region, and the PPAE. These are preliminary to solving spatial sampling. The problem formulation for spatial sampling was presented and introduced a utility function which is a linear combination of heuristic values. Before proceeding to the design of experiments needed to validate the approach to solution, the next chapter presents implementation details to aid in understanding the experimental design.

4 Software Implementation Details

4.1 Simulation of an Urban Environment

The goal of the simulation is to produce both simulated truth and target measurements within an urban environment for use in analyzing the tracker. The simulated environment has roads, stop lights and stop signs, cars, and miscellaneous effects of obscuration and parallax. The simulated urban environment was written in Matlab. Figure 17 shows a single frame of simulated video.

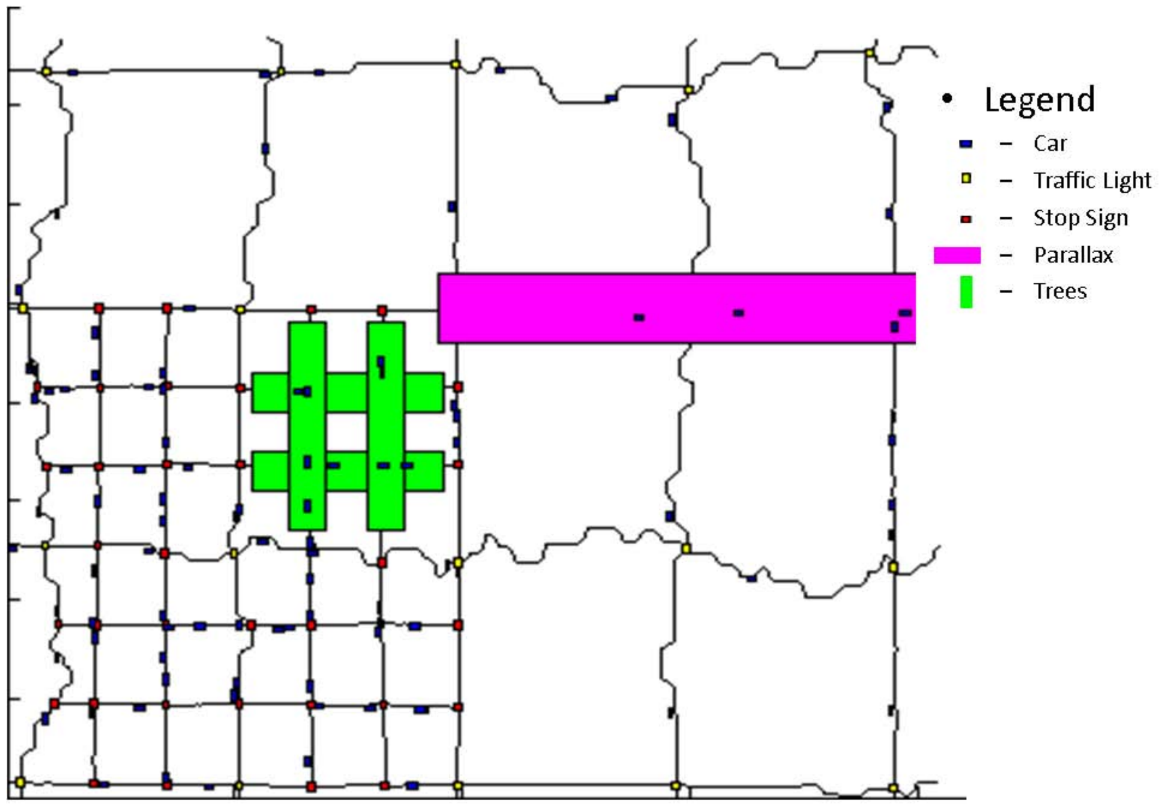


Figure 17 - A single frame of simulation video.

Figure 17 depicts 100 cars (blue, not drawn to scale) on roads (black, not necessarily straight lines) with traffic lights (yellow) and stop signs (red) at intersections. The green area represents trees and the magenta area represents parallax. More detailed descriptions are found below.

4.1.1 Resolution, Field of View, and Framerate

The resolution and field of view (FOV) chosen for the simulation is representative of currently available data such as the Columbus Large Image Format (CLIF) data collected by the Air Force Research Laboratory. Other resolutions and FOV are possible. The resolution used in the simulation is 2048 X 1536 pixels and other resolutions are possible. These values were chosen for computational ease and represent a standard image size. The FOV is such that a pixel is approximately 3 square feet and a vehicle is about 2 X 5 pixels. A larger FOV for the same resolution results in fewer pixels on target, which negatively impacts tracking. The framerate for the simulation is 30 frames per second (FPS) which is state-of-the-art for video of this size. A slower framerate of 1 FPS is used by downsampling the full simulation. At this time, we do not know the framerate anticipated operational HSI sensors, but expect it to be slower than 30 FPS. The framerate of 1 FPS was chosen to illustrate how a change in framerate will affect performance.

4.1.2 Roads

The roads are not all straight and their crookedness is randomly created, such that some roads are more crooked than others. There are 2-lane roads with a 25 mph speed limit and there are 4-lane roads with a 55 mph speed limit. The 2-lane roads are more closely placed and represent a residential area in the lower left of Figure 17. The 4-lane roads are farther spaced and represent a commercial area.

4.1.3 Traffic Lights and Stop Signs

There is a traffic light at each 4-lane intersection. For ease of implementation, all traffic lights are coordinated. All north-south roads are stopped at the same time while the east-west roads have a green light, then switching occurs. The duration of stop lights is one minute. There are no special lights for turning lanes. Cars that are turning will stop before turning if the light is red. Although these traffic lights do not fully model true traffic lights, the primary effect they have upon tracking is simulated through the stopping and starting of traffic, thus these simplifications should not affect fundamental tracking performance.

There is a stop sign at each 2-lane intersection and oncoming for the 4-lane intersection (the 2-lane vehicles stop, but the 4-lane vehicles do not). When a vehicle stops at a stop sign, it stays stopped from one to two minutes as determined by a uniformly random distribution.

4.1.4 Vehicles

There are 100 vehicles in the simulation at all times. When a vehicle drives out of the FOV (off a road exiting the scene), it is replaced by another vehicle that is randomly placed in the scene. The vehicles move using a constant acceleration model. The acceleration and velocity of each car will roughly obey the speed limits (+-20%), but will vary by individual car. They stop at all stop signs and red stop lights. They obey the prime rule of driving and do not occupy the same space at the same time as another vehicle. Vehicles will randomly change lanes and randomly turn at intersections. If a speeding car comes up behind a slower car in a 4-lane highway, it will change lanes and pass. The intent of this implementation is not to represent a comprehensive normalcy model of traffic flow, but to create challenging events for the tracking algorithm that also have a foundation in reality.

4.1.5 Obscuration

Vehicles are not always visible to an overhead observer. Sometimes the vehicles become obscured by driving under cover of trees or into shadowed areas. When vehicles are partially obscured, it is sometimes not possible to obtain a measurement of their location. In Figure 17 the green shaded area represents an area of obscuration. When vehicles are in this area, the probability of creating a measurement is arbitrarily set to 50%. HSI on the other hand presents an advantage in handling obscuration (although the exact effects are unknown). For the purpose of the simulation, the probability of creating a measurement using HSI in the green shaded area is arbitrarily chosen to be 75%.

4.1.6 Parallax

Parallax is the apparent motion of tall stationary objects caused from a change in camera angle. In our scenario, a plane flies in a circular pattern overhead and points the camera toward the center. Because of the changing camera angle, some roads are occasionally behind buildings and not visible. The net effect of the overhead plane is that tall buildings seem to exhibit a rotation compared to the ground. The purple shaded area of Figure 17 is an area of parallax. No vehicles are visible from within this region and no measurements are obtained. There are two simulated tall buildings in Figure 17 directly below the purple shaded region and bounded by the adjacent roads. The camera location for Figure 17 is at the bottom of the figure. The plane flies in a clockwise direction and circles the area in 6 minutes. Every 90 seconds different roads are obscured by parallax and previously parallax obscured roads become visible. The net effect is that parallax follows a clockwise pattern around the two tall buildings.

4.1.7 Context-Aided Tracking and Feature-Aided Tracking

The baseline tracker does not perform any context-aided tracking. The simulation, however is capable of providing detailed information about the roads (location and speed limits), buildings (location and parallax), and terrain (obscuration) for use in context-aided tracking design and testing. In real-world problems, this information is obtained through external sources or from historical tracking data.

While we did not explicitly perform feature-aided tracking, we did model the effects of feature-aided tracking, and thus simulated the performance of using a feature-aided tracker. The feature being modeled is HSI. With true feature-aided tracking, we would obtain HSI for the vehicle in question and use a classifier to identify the vehicle. Instead, we are simulating the output of the classifier and characterizing the output with the probability of correct classification (PCC). When we request HSI data from the sensor for a vehicle, no actual HSI is obtained. If the request is close enough to where the vehicle is, the simulation proceeds as if HSI had been obtained and uses PCC to determine if the correct classification is obtained. An ID is associated with the vehicle (correct or incorrect; based on PCC) and the ID is used in feature-aided track fusion.

Other methods could be employed within this simulation environment to represent other types of feature-aided tracking.

4.2 Tracker

This section describes the tracker in terms of the routines discussed in the background Section 2.2.3 (see Figure 3). It also discusses the association routine and the routines that fuse the ID obtained from the simulated HSI data in greater detail. The filter used for tracking is a linear Kalman filter using the white noise constant velocity model as described in Section 2.2.1.1.

4.2.1 Segmentation

No segmentation is explicitly used as the simulation provides measurements directly, thus simulating this routine as well. Measurements are created with measurement noise, but without clutter. Closely spaced vehicles in the simulation provide a natural source ambiguity commensurate with of clutter measurements without additional artificial clutter measurements. Missing measurements are created by the simulation through the obscuration and parallax information.

4.2.2 Gating and Association

Both a coarse square gate and a fine elliptical gate are used for association (see Section 2.2.3.2). The coarse gate is sized so as to enclose the fine gate (Figure 18). If a measurement is within the coarse gate, it is checked to see if it fits in the fine gate. The theoretical purpose of the coarse gate is to minimize computation of the fine gate and only apply it to those measurements that are close to the track under consideration. The coarse gate has no affect on the program results other than a theoretic improved computational efficiency, and everything would work the same with only a fine gate. No effort was used in this work to compare computational speeds with or without the coarse gate.

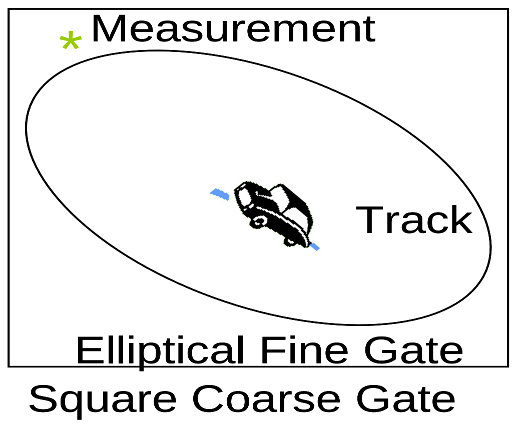


Figure 18 - A measurement that falls outside elliptical gate, but inside the coarse gate.

4.2.2.1 Association Decomposition by Gating

Since the association problem is NP-Complete, any advanced technique used to solve it will have scaling problems. To minimize this effect, the problem is first decomposed. The idea is to group all tracks and measurements that can possibly be associated and not group tracks and measurements that are too separated to be associated. Figure 19 illustrates how six tracks and eight measurements are decomposed into two separate problems of three tracks and four measurements each.

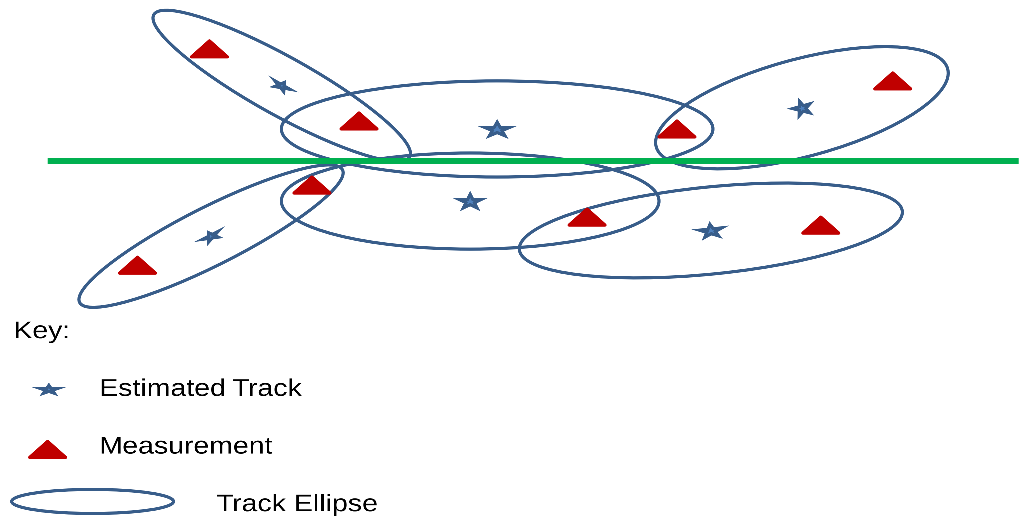


Figure 19 - Illustration of association decomposition. The tracks and measurements above the green line are associated separately from those below the green line.

The tracks and measurements above the green line are grouped together because of the measurements that fall within two ellipses. The same is true for the tracks and measurements below the green line. Even though the two central ellipses overlap, there is no measurement there; hence the two central ellipses will never compete for the same measurements.

The *Association_Decompostion* routine needs two sets of lists that are generated in the gating routine. Every track maintains a list of the measurements that fall within the gate. Likewise, every measurement maintains a list of the tracks to which it may be potentially associated and it is in the gate for. The coding is very compact, taking advantage of recursion:

```
Association_Decomposition(Measurement_List, Target_List);
{// Note:  For every Measurement contained in Measurement_List
Current_Measurement = next non-associated measurement from the Measurement_List
Add_Measurements(Current_Measurement, Measurement_List, Target_List,
Measurement_Sub_List,Target_Sub_List);
Association(Measurement_Sub_List,Target_Sub_List);
}

Add_Measurements(Current_Measurement,Measurement_List,Target_List,
Measurement_Sub_List,Target_Sub_List)
{
Measurement_Sub_List = Measurement_Sub_List + Current _Measurement;
Measurement_List = Measurement_List - Current _Measurement;
For every Target connected to Current _Measurement:
        If (Connected_Target is contained in Target_List)
        {
                Current_Target = next Target connected to Current_Measurement;
                Add_Targets(Current_Target, Measurement_List, Target_List,
                                Measurement_Sub_List,Target_Sub_List);
        }
}

Add_Targets(Current_Target,Measurement_List,Target_List,
Measurement_Sub_List,Target_Sub_List)
{
Target _Sub_List = Target _Sub_List + Current _ Target;
Target _List = Target _List - Current _ Target;
For every Measurement connected to Current _ Target:
        If (Connected_Measurement is contained in Measurement_List)
        {
                Current_Measurement = next Measurement connected to Current_ Target;
                Add_ Measurements (Current_ Measurements, Measurement_List,
Target_List,
                                        Measurement_Sub_List,Target_Sub_List);
        }
}
```

The *Association_Decompostion* routine starts by setting *Current_Measurement* to the first measurement in the *Measurement_List*. It then calls *Add_Measurements* which is described below. When *Add_Measurements* returns, the *Measurement_Sub_List* and the *Target_Sub_List* contain all the measurements and targets that are grouped together. Also, all the measurements and targets in the sublists are removed from the *Measurement_List* and *Target_List* since they have already been added to a sublist. The *Association* routine is unaltered by decomposition and is called to associate the sublists. The next non-associated measurement in the *Measurement_List* is set to *Current_Measurement* and the process continues until all measurements from the *Measurement_List* have been associated.

The *Add_Measurements* routine adds the *Current_Measurement* to the *Measurement_Sub_List* and removes it from the *Measurement_List* since it only needs to be added to a sublist once. It will then ensure that all tracks that the *Current_Measurement* is in the association gate for are added by calling the *Add_Targets* routine.

The *Add_Targets* routine is the logical complement to the *Add_Measurements* routine. It adds the *Current_Target* to the *Target_Sub_List* and removes it from the *Target_List* since it only needs to be added to a sublist once. It will then ensure that all measurements that fall within the gate of the *Current_Target* are added by calling the *Add_Measurements* routine.

The recursive coordination between *Add_Measurements* and *Add_Targets* creates a complete list of measurements and targets that will be associated together. All such decomposed lists are associated separately. The end result of the *Association_Decomposition* routine is identical to calling the *Association* routine without decomposition. The computational performance for *Association_Decomposition* is superior since it is faster to do several smaller problems than it is to do a single large problem.

4.2.2.2 Association by Lin-Kernigan 3-cut Method

Since the association routine is NP-Complete, it makes sense to use one of the finest heuristics for NP-Complete problems[55]. Lin-Kernigan (LK) is a heuristic used on the travelling salesman problem. The first step is to produce a potential solution. A greedy method is often used to produce a good first solution. I use the nearest neighbor method to produce the first association solution. The next step is to search for improvements. Conceptually, if we had a string showing the travelled path and cut the string in two places, then reversed one of the strings (re-attaching them to the opposite string where they were not originally) we would have a different solution. If the path is shorter, we have made an improvement and we keep it. If the path is longer, we revert back before the cut. The 2-cut LK performs all possible such 2-cuts and keeps all improvements along the way. The 3-cut LK is similar to the two cut, but performs all possible 2 and 3-cuts. The 2-cut method is of complexity is $O(n^2)$ while the 3-cut method is $O(n^3)$.

4.2.3 Initiation, Confirmation and Deletion

Any measurement that is not associated with an existing track will initiate the creation of a new track. Tracks are confirmed and deleted by the *M/N* method. For example, when a track has received 7 out of the last 10 possible measurements, it becomes a confirmed track and participates in the performance metrics. If, on the other hand, a track receives 3 or fewer measurements in the last 10 possible measurements, it is deleted. These values are somewhat arbitrarily chosen to ensure adequate tracking prior to confirmation and prevent early deletion of tracks. Other values or adaptive values could give better tracking results, however the use of a set value was chosen for consistent comparison while not overly increasing complexity.

4.2.4 HSI Exploitation

The simulated result of the HSI classifier is ID information for a selected TOI. This section describes how this information is used to enhance tracker performance. The HSI exploitation is accomplished through four routines which are discussed below. The first step is to select pixels according to the Utility function described in Section 3.6. The next step is to determine, for those measurements obscured by trees, if a HSI pixel detects the vehicle and so we add a measurement to the measurement list. We then perform an HSI association, and finally confirm identification (ID) of vehicles for which we received HSI.

4.2.4.1 Selecting Pixels

The pixels surrounding each TOI are given a value based on the utility function. We desire to select HSI pixels near enough to the target so that when an HSI measurement of the target is received, we actually get a pixel in the target rather than an arbitrary pixel. Selecting a bunch of adjacent pixels is a waste of resources. By spreading out the selected pixels, we are more likely to receive an HSI pixel of the target for the same resources as illustrated in Figure 20. Pixels are spread out from the estimated center pixel so they are no closer than 3 pixels apart. The resolution of the vehicles is such that a spread of 3 pixels will not miss the vehicle. The values of the pixels are sorted and the pixels with the highest value are used up to the limit of allowable pixels.

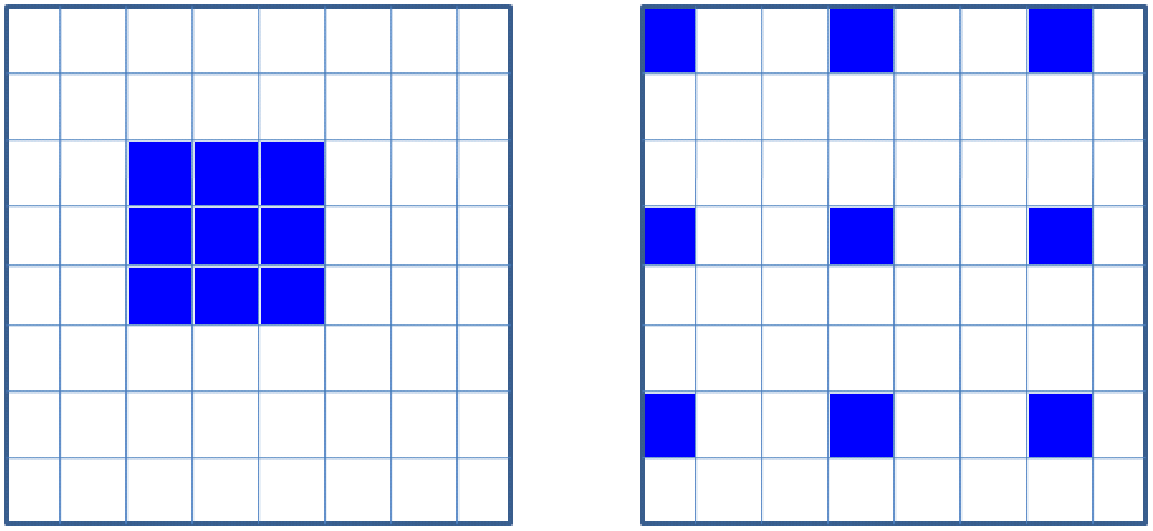


Figure 20 - Adjacent (left) vs. spread pixels (right).

4.2.4.2 Adding HSI Measurements

If a target is in an obscuration area, it has a lower chance of generating a measurement (50% chance in this simulation). HSI, on the other hand, has the ability to potentially receive a measurement from a partially obscured target. While the added benefit is quantifiably unknown, the simulation increases the chance of generating a measurement if HSI is used on the partially obscured target. Targets that are partially obscured have an additional arbitrarily selected 50% chance of generating a measurement if a pixel is selected near the target. This brings the total chance for generating the measurement to 75%. These partially obscured targets' measurements are added to the measurement list based on the HSI information.

4.2.4.3 HSI Hits and Association

In order to receive HSI information for a target, we must select a pixel near where the measurement for the target will be. If we select a pixel where the target actually is, we still might not get HSI information since the measurement for the target (and hence the relevant pixels) may be elsewhere due to noise inherent in the measurement process. An HSI hit results when a pixel selected within the sensor’s FOV is near enough to a measurement to infer we received HSI information for the target. When a selected pixel is within 2 pixels of a measurement, we receive HSI information for the target and we have a HSI hit. Even though we have a HSI hit for a target, we still might not classify it correctly. After a HSI hit, the measurement is given the correct ID of the target according to the probability of correct classification (PCC). A random number is generated and compared to the PCC. If the random number is less than or equal to the PCC, the correct ID is used. If it is greater than the PCC, an incorrect ID is used. In order to maximize confusion (thus creating the worst-case possible incorrect classification), each target is paired so that if two paired targets both get an incorrect classification, they will both get each other’s classification.

HSI association occurs after a HSI hit has been established. The only candidates for HSI association are those that are kinematically close (in terms of distance between a target and a measurement) and fall within the association gate of the track (see Section 2.2.3.2). If no tracks are kinematically close to a HSI hit, no HSI association occurs. The HSI ID of the measurement is compared with the ID (obtained from previous HSI hits) of the nearby tracks. In order for HSI association to occur, the IDs must match. If there are multiple tracks with a matching ID, the first tie-breaker is confirmation (See Section 2.2.3.4). If only one of the matching tracks has been confirmed, it is selected to undergo HSI association. The second tie-breaker is track order (which is related to track age, i.e., oldest track). HSI association pre-empts the regular association routine. When a measurement and track are HSI associated, they are forced to be associated by the kinematic association routine.

4.2.4.4 HSI ID Confirmation

The purpose of HSI ID confirmation is to use the HSI measurement to generate and correct the ID information for each track. One benefit is that it counters the ill effects of track swaps by correcting the ID information based on the HSI ID. The ID information is changed when a track is associated with two successive (in terms of HSI hits, not time) HSI hits where the ID information differs. An HSI association breaks up the succession since in that case the ID information matches.

A concern for this step is the false confirmation rate. At first glance, it would seem as if the ID of a target would be incorrectly changed at the rate of (1-PCC)*(1-PCC). In the case of PCC = 70%, this amounts to a false confirmation rate of 9%. However, the kinematic association also plays a role and reduces this rate. This rate could be further reduced by increasing the number of successive HSI hits required.

4.3 Multi-objective Algorithm NSGA2

NSGA2 is used for filter tuning and to optimize the weightings in the utility function (see Section 3.15). NSGA2 is described in Section 2.2.6.4.1. It is readily available and freely downloadable. It is used without modification. This section discusses the parallelization which is extraneous to NSGA2 and parameters used in NSGA2.

Since a single evaluation of 10 minutes of 30 fps data for 10 Monte Carlo runs can take an hour (note this is faster than real-time execution), and in order to compute 100 evaluations for 100 *generations*, the total computational time is 10,000 hours. Parallelization seemed a necessity. NSGA2 itself is not parallelized, but we parallelized the evaluation of *individuals*. For parallelization, we have 16 processors available. For each *generation*, each processor is given 6 *individuals* to evaluate using the tracker (for a total of 96 *individuals* per *generation*). Each processor reports their results and NSGA2 proceeds as normal to select the next *generation* for evaluation.

The parameters used in NSGA2 are recorded for completeness:

Table 2 - NSGA2 Parameters

```
nobj = 2;// the number of objectives
nreal = 3;// the number of real variables; (2 for tuning)
min_realvar = various;
max_realvar = various;
// The min and max values are set according to the task and framerate being tested. For
finding the C values of the utility function, the min is 0 and the max is 1. For filter tuning
(where these values represent q and r), the min and max are set around the tuning results
and were narrowed over successive runs
ncon = 0; // the number of constraints
pcross_real = 1.0; // the probability of crossover of real variable
pmut_real = 0.05;//probablity of mutation (1/nreal) 5%
eta_c = 10; // distribution index for crossover (5-20)
eta_m = 10;     // index for mutation (5-50)
nbin = 0; // number of binary variables
```

4.4 Chapter Summary

This chapter described the urban simulation, tracking algorithm, and NSGA2 in enough detail to understand how they fit together. These are used throughout the experiments which are presented in the next chapter.

5 Design of Experiments

Many experiments are needed for this effort, and a number of experiments explore the validity of the PPAE equations. Another set of experiments revolve around tuning the Kalman filter, since it needs to be tuned prior to operation in the utility function experiment. Finally, the last set of experiments use the utility function and demonstrate the performance with various parameters.

5.1 PPAE Experiments

5.1.1 Validation of 1-D PPAE Equation

In order to explore the nature of the 1-D closed-form PPAE Equation (47) and provide validation, we use a MATLAB program that picks millions of instances of measurements Z_1 and Z_2 and experimentally derives the numeric solution. In the experiment, $Z_1 \sim N(0,1)$, $Z_2 \sim N(x,\sigma^2)$. The parameters x and σ are varied in order to visualize the effects of changing the distance between targets or changing the standard deviation of the targets' uncertainty. The parameter x is varied from 0.1 to 2.0 in steps of 0.1, and σ is varied from 0.4 to 2.0 in steps of 0.4. For each value of x and σ a million instantiations of Z_1 and Z_2 are generated. The percentage of those that cause an association error is the experimentally derived PPAE for that value of x and σ. One hundred such experimentally derived values are obtained since x and σ are in a nested loop. A plot of PPAE versus x for each value of σ illustrates both the effect of changing x and σ.

5.1.2 Visualization of Limits of Integration for 2-D Convolution PPAE

This experiment is performed in order to understand the nature of the limits of integration of the 2-D PPAE Equation (48) which is repeated for the convenience of the reader.

$$PPAE = \int \int_{f2(\vec{\xi})}^{f1(\vec{\xi})} PDF(Z_1, \vec{\xi}) PDF(Z_2, \vec{\psi}) d\vec{\xi} d\vec{\psi} \tag{54}$$

The innovation covariances and state estimate of positions are given for two targets. The targets use ellipse parameters of a = 3, b = 1, (x_0, y_0) = (0,0) and φ = 45 for the first target and a = 3, b = 1, (x_0, y_0) = (2,0) and φ = 135 for the second target. These two ellipses are depicted in Figure 21, which illustrates how these arbitrary values result in a significant level of target ambiguity.

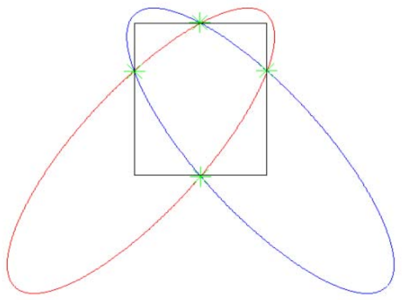


Figure 21 - Visualization of target ephemeris.

A realization of Z_1 is also given for a location in the overlapping region of the innovations of the targets. A thousand random measurements are generated for Z_2. If the instance of Z_2 causes an association error, the location is marked in magenta. If Z_2 and Z_1 are approximately equal, Z_2 is marked in black. In this way, the approximate lines of equality are seen and regions of association error are noted for a given Z_1. The experiment is repeated with different realizations of Z_1. Values of Z_1 vary from (0.75, 0.75) to (1.25, 1.25) in steps of 0.25 for a total of 9 points. The central point (1,1) is in the center of the overlap region and the 8 adjacent points surround it.

5.1.3 Determining the Area Probability and Joint Probability that Two Targets Overlap

This experiment validates and compares the numeric approximation of determining the probability a target is within an arbitrary area by small squares Equation (34), by small arcs Equation (37), and the joint probability that two targets are within an arbitrary area Equation (38). While any arbitrary area can be used to demonstrate these methods, the overlapping area of two ellipses is of primary concern to us since it allows us to calculate the joint probability of targets being close and can be used as a measure of potential association error. Five ellipses representing the innovation of five targets are presented which in combination give ten overlapping areas. The probabilities of targets being in the overlapping areas are calculated by both the small squares Equation (34) and the small arcs Equation (37). To further validate the approach, ten thousand random target locations are generated, and the percentage of those locations falling within the arbitrary location is reported as a percentage, hence an approximation of the probability through sampling. These values are then used to calculate the joint probability that the two targets are in the overlapping region by Equation (38).

The parameters used to satisfy the ellipse Equation (21) for 5 ellipses are shown in Table 3. These parameters define the five ellipses which in combination give ten overlapping areas for use in calculating the joint probability of two targets overlapping.

Table 3 - Equation parameters for 5 ellipses

Ellipse#	a	b	x_0	y_0	φ(deg)
1	3	1	0	0	45
2	3	1	2	0	0
3	3	1	0	0	0
4	4	3	0	0	135
5	3	1	1	1	0

5.1.4 Comparison of Approximation of PPAE, Numeric Calculation of PPAE, and Joint Probability that Two Targets Overlap

This experiment builds on the results of the prior experiment. PPAE is numerically calculated from Equation (51) for the same 10 ellipse combination pairs as in Table 3 with both *dx* = 0.5 and *dx* = 0.25. PPAE is also approximated for the same 10 ellipse combination pairs through sampling by generating 10,000 pairs of measurements and totaling those that would cause an association error. These results are then compared to the joint probabilities found in the prior experiment. This experiment validates the numeric approximation of PPAE and illustrates the difference between PPAE and the joint probability that two targets overlap.

5.1.5 Comparison of Numeric Calculation of PPAE vs. Actual Results in a Tracker

The theory under which PPAE is developed includes the assumption the innovation represents the true probabilities for the future measurement. In reality, track error, missed measurements, prior association errors and other challenges make the assumption suspect. In addition, interactions with more than two targets further complicate the PPAE. This experiment is designed to compare actual association errors inside a tracker to those computed by PPAE.

The tracker is run using the first minute of the simulation and all 10 Monte Carlo runs. Tuning values for the tracker are q = 2 and r = 5. These values were chosen after tuning by hand. An exact tuning is not deemed necessary since tracking performance is not being measured. The measurement noise level is arbitrarily set to 5 and the framerate is 30 fps. At each timeframe, PPAE is computed for all existing tracks. If there are multiple interactions, PPAE is summed pairwise to calculate a total PPAE for that track. Tracks with similar values of PPAE are binned together. The first bin is a PPAE of 0. The second bin is a PPAE between 0 and 5%. Further bins increase in increments of 10% with the third bin between 5% and 15%. If a track produces an association error (see Equation (16)), both the numerator and denominator for the corresponding bin is increased by 1. If a track does not produce an association error, the denominator for the bin increases by 1, but the numerator does not change. This allows us to compare our PPAE to association error. Ideally, when we predict a PPAE between 10% and 15%, the resulting division for the bin should be between 10% and 15% of actual association error.

The experiment is repeated as above with tuning values of $q = 2$ and $r = 9$. This allows us to illustrate how changes in tuning values affect the correlation and accuracy of the PPAE.

5.2 Tracker Tuning

Two types of experiments are conducted regarding tuning. The first seeks to quantify the need or lack of need for Monte Carlo (Monte Carlo) runs in tuning a multi-target tracker using a multi-objective algorithm. The rest of the experiments may be better termed a procedure. The tracker is tuned for use in further experiments.

5.2.1 Monte Carlo Tuning Required for Multi-target Tracking

The purpose of the experiment is to quantify the need or lack of need for Monte Carlo (MC) runs in tuning a multi-target tracker using a multi-objective evolutionary algorithm (MOEA). The question being addressed is "Rather than conducting tuning with Monte Carlo over all scenario data sets, how close to optimal would one be if only a single MOEA tuning over one data set is used?" Note that we use a parallel/distributed computational environment for efficiency in our experiment.

The benchmark scenario available for the experiment is 10 stochastic sets of data with 100 targets to be tracked. The tracker is tuned using a Monte Carlo approach and a MOEA, the NSGA-2. The q and r *individual* tuning parameters are scored for each algorithm using the objectives of fraction of missed targets (FMT) and association error (AE) (see section 2.2.3.5). A Pareto front (PF) for each of the individual NSGA-2 10 data sets is computed. Also, a Pareto front for the Monte Carlo data using all 10 data sets is produced for comparison. In order to answer the question posed, we must proceed to select an "optimal point" as if we had only a single run over a randomly selected data set and compare that point to the solution based on Monte Carlo over all 10 data sets. A median Pareto front point is selected for each of the NSGA-2 individual data set runs that is “closest” to the Monte Carlo Pareto front. The tuning parameters that produced the selected median points for each of NSGA-2 individual runs are evaluated using the two objective evaluations. Since both FMT and association error are percentages (although of a differing nature), 2-D Euclidean percentage distance is used to represent the closest distance from the Monte Carlo Pareto front for each NSGA-2 data set scenario execution.

For this problem scenario, each solution consists of two real valued variables q and r which are used to scale Q_d and R respectively to produce the tuning parameters. The NSGA-2 population size is set at 96 since we are running 16 parallel processors each evaluating 6 tuning solutions. A *generation* consists of running the tracking program using the values of Q_d and R derived from each solution in the *population* on a given set of data (i.e., the scenario). NSGA-2 is allowed to run for 100 generations. Since there are 10 stochastic runs of the scenario, 11 PFs are produced, one for each run and one using a Monte Carlo scoring. Note that the NSGA-2 is run ten times per data set scenario to produce an average known Pareto front. In scoring an *individual* solution for the NSGA-2, the NSGA-2 is run ten times for each scenario data set. The 2-D Euclidean percentage distance measure is evaluated and scored for each run. For the Monte Carlo Pareto front and the NSGA-2, the distance score is the average score over the NSGA-2 individual runs of each data set scenario.

It would be easy and incorrect to think of the Monte Carlo Pareto front as the average of the other NSGA-2 10 Pareto fronts. This would only be true if the same set of *individuals* produces every Pareto front, which is not the case. Once an average Pareto front is generated for a NSGA-2 run per scenario, a “median point” is selected and the *individual* which produced the point is used as the operating point. For each NSGA-2 individual run average, the median point of the Pareto front is selected as “closest” to the Monte Carlo Pareto front per the Euclidean distance. Selected *individuals* are the ones that would have been used for tracker tuning if we did not use Monte Carlo. By comparing them to the Monte Carlo Pareto front, we can determine the value of using Monte Carlo.

5.2.2 Tracker Tuning for Spatial Sampling

Tuning the tracker for spatial sampling is more accurately termed a process rather than an experiment. This step precedes spatial sampling since it is desirable for the tracker to have optimal performance for the baseline tracker without spatial sampling or HSI exploitation. The tracker is tuned four times using the objective functions of FMT and IDE (See Sections 2.2.3.5.1 and 2.2.3.5.2). The tracker is tuned for a framerate of 30 fps and 1 fps. The measurement noise at both framerates is set at three and ten. The tuning uses Monte Carlo scoring for the entire 10 minutes of the scenario. These parameters were chosen and used for consistency in the experiments discussed in the next section. After the tracker is tuned, the tuning values are reported and used in the baseline tracker throughout further tests.

5.3 Spatial Sampling

After discussing some overall design considerations, we list the factors included in the experiment which are believed to affect performance. The experiment is then presented along with the purpose of conducting the experiment. Thereafter, the factors which affect performance that are not included are discussed theoretically and related to the experiment.

5.3.1 Design Considerations

Since the tracking system being tested is not an operational system, a concern is how can the results of the test apply under more realistic conditions with an entirely different system? In order for results to be relevant, they must either apply directly to a relevant system under consideration, or be conducted upon a system in such a way that the results can be applied indirectly to another system. While we have spent a great deal of effort to create a realistic environment for the test, the overall design consideration is to produce a lower bound for performance. These tests are indirectly related to a real operational system, and performance of the real system should be better than those presented in this experiment. Additionally, future implementers should be able to have some idea of how much better an operational system performs by extrapolating from this experiment.

5.3.2 Factors that Affect Performance Included in Testing

The parameters which are varied in the test are the framerate (1fps or 30fps), the measurement noise (3 or 10), the number of HSI pixels available (10, 100, or 1000), the performance of the HSI classifier (PCC = 95 or 70), and the constant values used in the utility function which are varied by NSGA-2 and manually set to show each component in the utility function by itself and in equal proportion. The reasoning for each of these parameters and how they affect the baseline performance is discussed below.

The baseline performance of the tracker with no HSI is affected by the framerate of the measurements. Naturally, the higher the framerate, the better the baseline tracker performs. A certain level of performance by the baseline tracker is required for the HSI data to be useable. If the target estimate is too far removed from the truth, then the attempt to receive an HSI measurement based on the target estimate will fail, thus rendering the HSI data nearly useless. The baseline tracker is tuned to optimally perform based on the framerate. Framerates of 30 fps and 1fps are tested in order to show how performance is affected with a higher and lower framerate.

In order for HSI data to be associated with a target, the selected pixel must be within the resolution size of the target (which for this system is about three pixels for a vehicle). Early proof-of-concept experiments showed that with ten HSI pixels available, a measurement noise of ten pixels produced very little or no improvement compared to the baseline, but with a noise of three pixels, the improvement was observable. For this reason, measurement noises of three and ten pixels are used.

After establishing the baseline with no HSI, the performance of the tracker with HSI will depend heavily on the number of HSI pixels available. If HSI pixels were available for every pixel and every frame, we could expect near perfect performance. The number of HSI pixels available determines a point between the baseline and perfect performances. The number of HSI pixels tested is 10, 100, and 1000 along with perfect mode where HSI is available for every pixel every frame. These values are chosen to illustrate how varying the number of pixels affects performance.

With HSI, the ability of the classifier to correctly identify the target also affects performance. Classifier performance depends not only on the classifier algorithm used, but also on the quality of the data gathered by the sensor. The quality of the data is further affected by design considerations of the sensor such as resolution and framerate. While it is not possible to know how a classifier will perform until the sensor is built, it is possible to show how a change in classifier performance will affect tracking performance. As the PCC increases, we expect performance to increase. A PCC of 95 and 70 is used because these values are the expected extremes of classifier performance.

All of these testing parameters are tested in full combination with each other. This enables us to perform tradeoff analysis among the various parameters.

5.3.3 Validation of the Utility Function

This test is performed to validate the utility function under the parameters of framerate, the measurement noise, the number of HSI pixels available, and the performance of the HSI classifier as discussed above. In order to have a basis of comparison, the tracker is tuned using the 10 minute Monte Carlo simulation and the performance is noted for the metrics of FMT and IDE. This is denoted as the baseline performance without HSI. An upper performance, which we term perfect HSI, is further established by forcing every measurement to be a HSI hit (simulating the results of every pixel being available at every timeframe). In order to more fully substantiate the use of the synergistic utility function, we need to show that the sum is greater than its parts. Each of the four components of the utility function (association $U_{ij}^{A}(t)$, model age/time $U_{ij}^{\Im}(t)$, measurement $U_{ij}^{M}(t)$, and default $U_{ij}^{D}(t)$ from Section 3.7) is operated independently by setting the corresponding constant value to 1 and all other constant values to 0. We have termed these four independent modes PPAE, periodic poling, missed measurements, and equal dispersion. The manner in which these independent modes disperse resources is discussed below.

The first utility component is PPAE. PPAE distributes the HSI pixel resources purely on the PPAE heuristic. A track that is completely isolated from other tracks will receive no HSI pixels. In order for a track to receive pixels, its ephemeris must overlap that of another track. The track with the highest PPAE will receive the most HSI pixels.

The next independent utility function mode is periodic poling. Periodic poling is the result of using the model age utility independently. A track receives HSI pixel resources based on how long it has been since it has not received an HSI hit. Initially, all tracks are equal and receive an equal amount of pixel resources, but as a track receives an HSI hit, it subsequently receives no resources until sufficient time lapses. This time delay until a track is revisited is why we call it periodic poling. Perhaps unlike pure periodic poling, if resources are allotted and a HSI hit does not happen, the track will still receive resources until an HSI hit does occur.

The third independent utility function mode is missed measurements. The missed measurements mode distributes the HSI pixel resources purely based on the number of measurements a track has missed in the last 10 timeframes. A track that has not missed any measurements will receive no resources. Those tracks that have missed lots of measurements will receive the most resources.

Equal dispersion is the last independent utility function mode. In this method, every track receives an equal number of available HSI pixels, thus equal dispersion is an apt name. This is perhaps the most natural approach by requiring little effort/optimization and provides a good basis of comparison for the other methods. The tie-breaker for resources is distance of track estimate from integer or floor values which is due to the inherent nature of the utility function. This amounts to a random method since these values change every timeframe.

We demonstrate the synergistic utility function by equally weighting (C = 0.25) the components in the utility function. Under this method, every track competes for HSI resources based on how they are in all aspects of the utility function. A track will receive the most resources if its summation of the utility function is the highest. The equal dispersion component value ensures some resources are distributed to each track when there are adequate resources available.

Finally, the components are optimally weighted by using NSGA-2 to find the appropriate weightings of the constant values in the utility function. This optimal weighting is compared to equal weighting to determine the potential improvement through optimization. The optimally weighted utility function is a more realistic upper performance bound than the perfect HSI performance bound, since it is unlikely that every pixel will be available as a source for HSI. By conducting individual component testing and synergistic utility function testing, we are able to compare the utility function to other approaches.

5.3.4 Factors that Affect Performance Not Included in Testing

The numbers of variables being tested geometrically increases the amount of simulation required for a thorough and complete test. There are a number of parameters which affect performance which we have willfully neglected in order to make the problem tractable and testable. Rather than ignore them completely, we wish to provide guidance on expected behavior as these parameters are varied. These factors are tracker performance, simulation factors (# of cars, changes in amount of parallax or obscuration, resolution), and tuning parameters.

We chose to use only a single target tracker. In keeping with the design goal of providing a lower performance bound, it has the fewest features available and uses perhaps the crudest methods. There are many possible improvements. Suggested enhancements include filter enhancements (multi-model filter, improved modeling, non-linear filter, adaptive filter), association enhancements (multi-hypothesis testing), and perhaps even some improvement in the confirmation and deletion (we never optimized or tuned the values of M/N = 7/10 for confirmation and 3/10 for deletion; we selected them and kept them). A commercially available tracker should outperform this tracker. How would an improved tracker affect performance? We will look at the change in measurement noise from 10 to 3 as if it were an improved tracker to analyze what may happen as a commercially available improved tracker is used instead.

We also chose to use a single simulation, and tracker performance is likely to be affected by simulation factors. If there are more cars, more parallax or obscuration, or lower pixel resolution, the tracker performance decreases. Once again, we can expect the performance change based on these simulation factors to behave as if the tracker or noise levels are changed. Additionally, the number of cars in the simulation affects performance of the tracker due to scaling issues. It can, for example, have a nearly geometric computational performance affect. Besides the change of noise (which should somewhat model this), we will look at the change in HSI pixels available relative to the number of cars to see how this simulation factor should affect performance.

Finally, we chose a single tuning (per noise level and framerate) to test with. We optimized this tuning using NSGA-2, however we probably did not need to use such a precise tuning. A rough tuning would have probably served our purpose. We can view an improved tuning in general as an improvement in the tracker as described above.

5.4 Chapter Summary

The purpose of all tests described in this chapter is to validate the utility function. Experiments that validate PPAE are needed since it is a component of the utility function. The baseline tracker is tuned so as to have a tuned tracker for use in validating the utility function. These sets of experiments are described in great detail. The utility function experiments themselves are perhaps more broadly described. The large number of parameters used in combination makes it impractical to give a detailed description of each experiment. Hopefully, we have conveyed the purpose in varying each of the selected parameters. As the reader goes forward into the results chapter an understanding of these parameters should enable an understanding of the detailed experiment presented by the results.

6 Results, Analysis, Conclusions and Future Work

The results, analysis, conclusions, and future work are presented in the same order as they were presented in the experiments section with PPAE first followed by tuning and spatial sampling.

6.1 PPAE

6.1.1 Validation of 1-D PPAE Equation

The five curves of Figure 22 show the effects of varying the distance between targets (x) from 0.1 to 2.0. Each curve has a different σ with the blue curve $\sigma = 0.4$ and each higher curve having an additional 0.4 ($\sigma = 0.4 + 0.4$) until the top curve (violet) has $\sigma = 2.0$. We used Equation (47) to produce various points (not included) along the curves and validated the results. Variations in σ for Z_1 should behave similarly since the same problem could be reformed with Z_1 and Z_2 relabeled alternately, thus the variation in Z_1 will have the same effect as a variation in Z_2.

Equation (47) for the 1-D PPAE confirms our intuitive insight that the 1-D PPAE depends on the distance between the targets and the standard deviations. Figure 22 further illustrates how the PPAE depends on the distance between the targets and the standard deviations. Figure 22 appears linear in the regions where PPAE is high. When PPAE is low, the linearity breaks down, however it is less important to be accurate in those regions since there is a small probability of an association error. It seems reasonable that a linear equation will attain a close approximation for PPAE. A first-cut is shown in Figure 23.

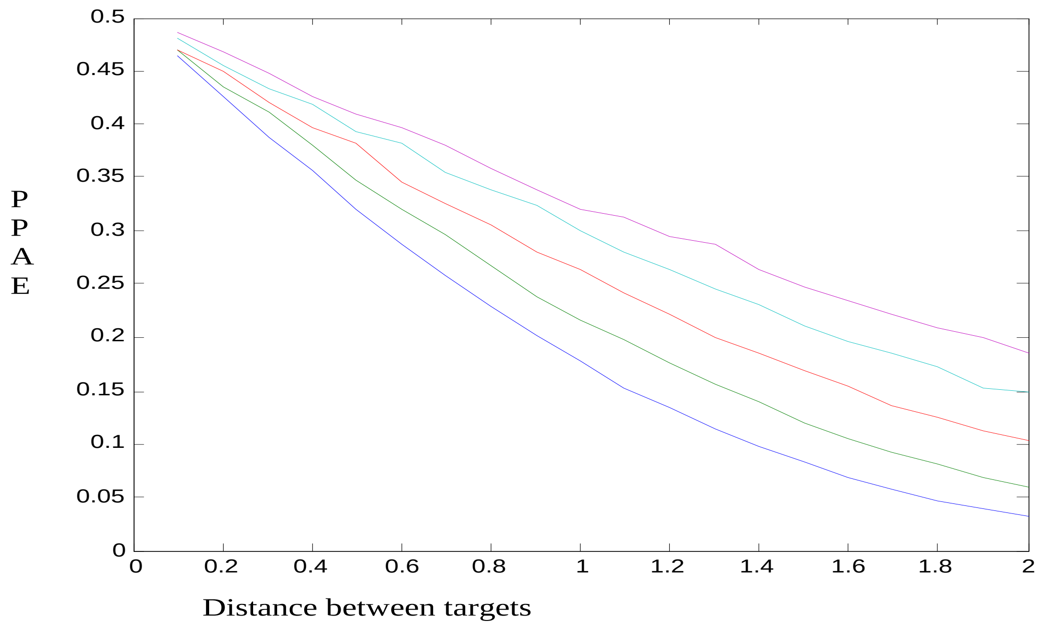


Figure 22 - PPAE vs. difference between means with varying distances and means.

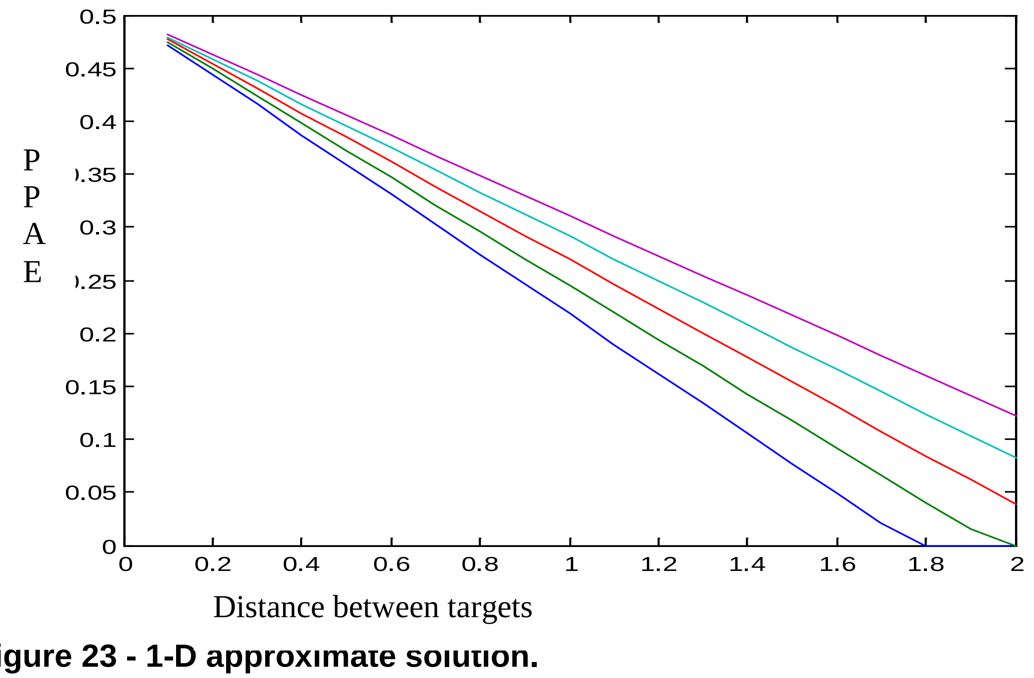


Figure 23 - 1-D approximate solution.

The equation for the above lines is:

$$y(x,\sigma) = .5 - (.4/\exp(\sigma/4))*x; \tag{55}$$

where σ is the sum of standard deviations for the Z_1 and Z_2.

It is possible that the 1-D approximation will correlate well with the 2-D problem. The challenge of such an approach is to determine what single mean from each of the targets' 2-D means to use in Equation (55). A first order attempt would be to average them. With such an approach, the error would depend on the angle between the two targets relative to the major and minor axes and how that angle varies with the 2-D means. A more refined approach would be to determine a mean between the two 2-D means based on this angle. Thus, if the angle between the two targets was aligned with the major axes, then the means associated with the major axes would be used, and likewise if the angle is aligned with the minor axes, the means associated with the minor axes would be used with variations between.

The 1-D approximation solution was not pursued since a numeric 2-D solution was found that can be calculated in real-time. It is presented here as a potential approximate solution with better computational performance than the 2-D solution.

6.1.2 Visualization of Limits of Integration for 2-D Convolution PPAE

The mosaic of Figure 24 shows the limits of integration for Equation (48). In each plot, the ellipses represent a 99.99% probability of a measurement being within the ellipse. The black rectangle encloses the overlapping region. The realization for the red ellipse (Z_1) is given and represented by a red asterisk. The center plot shows the limits of integration for the given Z_1 of (1,1) which is in the center of the overlapping area. The surrounding plots correspond to the change in Z_1 with (0.75, 0.75) in the bottom left and (1.25, 1.25) in the top right. The magenta points form an area where association errors occur and the black points show the curve of approximate equality where it is close to an association error occurring.

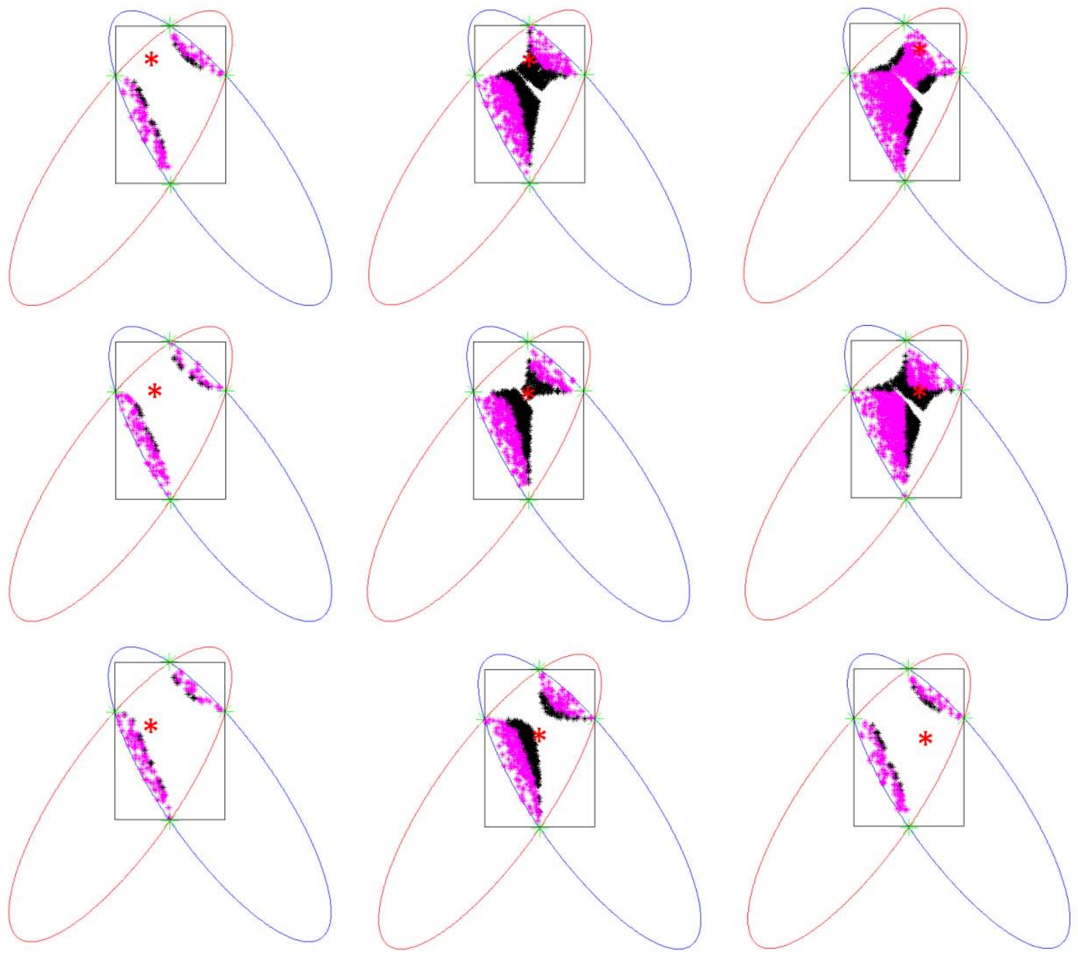


Figure 24 - Visualization of limits of integration for PPAE with varying values for measurement Z_1.

The limits of integration for Equation (48) appear hyperbolic. If association were allowed outside the elliptical gate, the black lines would go on and form an asymptote. Future work could be to determine the nature of these limits of integration and find a closed-form solution. The PPAE for a given measurement Z_1 is the summated product of the probabilities for Z_1 and Z_2 (the magenta points). Therefore, the amount of magenta is related to the PPAE where more magenta will indicate a higher PPAE. However, specific instances may differ since the location of the points is important. In general, Figure 24 illustrates that PPAE is smaller when the Mahalanobis distance from the center of the red ellipse to Z_1 is smaller. This makes logical sense since association errors are less likely for measurements that are statistically closer to the estimate and more likely for measurements that are farther away from the estimate. Comparing the center plot of Figure 24 with that of the upper-right corner is interesting. A switchover occurs where the center area is devoid of association errors (center plot) to where the center area is full of association errors (upper-right corner plot). In investigating the phenomenon, one could do more plots similar to Figure 24 and create an animation near the crossover point. I expect the hyperbolic limits of integration to devolve to an ellipse or a line. It also seems the phenomenon is related to large Mahalanobis distances. Perhaps there is a distance for which the switchover occurs. Future work could be to examine this phenomenon. While I believe the phenomenon can be explained using a closed-form solution, the reverse may also be true. Exploring this phenomenon may provide valuable insight leading to the discovery of the closed-form solution.

6.1.3 Determining the Area Probability and Joint Probability that Two Targets Overlap

The probability of a target being within the arbitrary boundary defined as the overlapping region between two ellipses is calculated for 10 overlapping regions by small squares, small arcs, and random generation. The ellipse numbers refer to those listed in Table 3 with, for example, 1-2 being the overlapping region of ellipse #1 and ellipse #2 of Table 3. The first multiplicand reported is the probability of a target being within the first listed ellipse number and the second one is the probability of a target being within the second listed ellipse number. The product is the joint probability of both being in the overlapping region. The *dx* and *dy* terms for the small squares method is 0.5. The *dr* and *dφ* terms for the small arcs method are 0.5 and 1 degree respectively.

Table 4 - Joint Probability Computations *dx* = *dy* = *dr* = 0.5, *dφ* = 1 Degree

Ellipse #s	Joint Probability by Squares	Joint Probability by Arcs	Joint Probability by Random
1-2	.6108*.2778 =.1696	.5726*.2989 = .1712	.5598*.3250 = .1819
1-3	.7905*.7637=.6037	.7512*.7512=.5642	.7624*.7579=.5778
1-4	.8903*.5965=.5310	.9967*.4566=.4552	.9894*.4561=.4513
1-5	.4921*.6892=.3391	.4675*.6603=.3087	.4399*.7052=.3102
2-3	.6980*.6980=.4872	.7205*.7205=.5191	.7659*.7278=.5574
2-4	.9035*.4710= .4256	.8839*.3017=.2667	.89728.3245=.2911
2-5	.4989*.4759=.2374	.3745*.3745=.1403	.4516*.2737=.1236
3-4	.9937*.6931=.6471	.9970*.4741=.4726	.9887*.5031=.4974
3-5	.4759*.4989=.2374	.3745*.3745=.1403	.4623*.3385=.1565
4-5	.4605*.9191=.4232	.3291*.9626=.3168	.2701*.9561=.2582

The same probabilities are calculated with the *dx* and *dy* terms for the small squares method equal to 0.25 and the *dr* and *dφ* terms for the small arcs method as 0.25 and 1 degree respectively. The random method results above are repeated for convenience.

Table 5 - Joint Probability Computations *dx* = *dy* = *dr* = 0.25, *dφ* = 1 Degree

Equation #s	Joint Probability by Squares	Joint Probability by Arcs	Joint Probability by Random
1-2	.5826*.2420=.1410	.5546*.2916=.1617	.5598*.3250=.1819
1-3	.7460*.7491=.5589	.7265*.7265=.5278	.7624*.7579=.5778
1-4	.8970*.5915=.5306	.9970*.4233=.4220	.9894*.4561=.4513
1-5	.4501*.6896=.3104	.4690*.6525=.3060	.4399*.7052=.3102
2-3	.6848*.6848=.4690	.7255*.7255=.5263	.7659*.7278=.5574
2-4	.8575*.4497=.3856	.8869*.3068=.2721	.8973*.3245=.2911
2-5	.4246*.4113=.1746	.3732*.3732=.1392	.4516*.2737=.1236
3-4	.9129*.6451=.5890	.9970*.4635=.4621	.9887*.5031=.4974
3-5	.4113*.4246=.1746	.3732*.3732=.1392	.4623*.3385=.1565
4-5	.4229*.8972=.3794	.3370*.9648=.3252	.2701*.9561=.2582

Both the small squares method and the small arcs method of computing the joint probability that two targets overlap is compared to the random sampling method for both *dx* and *dr* = 0.5 and *dx* and *dr* = 0.25 by looking at the correlation coefficients between them in Table 6.

Table 6 - Correlation Coefficients for Small Squares and Small Arcs Compared to Random Sampling

	Correlation Coefficient
Small squares ; dx = 0.5	0.89588
Small arcs; dr = 0.5	0.987825
Small squares ; dx = 0.25	0.918195
Small arcs; dr = 0.25	0.984018

Another way of comparing the various methods is by looking at the average and standard deviation of the difference between them.

Table 7 - Average and Standard Deviation of Difference for Small Squares and Small Arcs

	Average Difference	Standard Deviation
Small squares ; dx = 0.5	0.08609	0.053793
Small arcs; dr = 0.5	0.02087	0.017042
Small squares ; dx = 0.25	0.06041	0.040247
Small arcs; dr = 0.25	0.02890	0.018373

Both Table 6 and Table 7 show that the small squares method and the small arcs method of computing the joint probability work well when compared to the random sampling method thus validating both these approaches. The small squares with dx = 0.25 agreed more closely with the random sampling method as indicated by both a higher correlation coefficient and a smaller average divergence and standard deviation than with *dx* = 0.5. This was the expected result since the overlap error decreases as *dx* decreases. Surprisingly, the small arcs method with *dr* = 0.5 agreed more closely with the random sampling method as indicated by both a higher correlation coefficient and a smaller average divergence and standard deviation than with *dr* = 0.25. The difference is very slight. We initially suspected the cause is not that the *dr* = 0.5 is more accurate as the data might suggest, but rather that since we are making a comparison to an imperfect random sampling method and the result is due to random sampling errors. In examining this possibility, we took the worst-case error using ellipse 1 and ellipse 3 and performed random sampling 10 times with 10,000 samples. The average probability of association error was .5785 (vs. .5778 in the original data) with a standard deviation of 0.004. While this validates the use of random sampling as a basis for comparison, it removes one of the more likely causes why *dr* = 0.5 is better than *dr* = 0.25. Since the cause is not the error in random sampling, we must look at the contributing errors in the integration process itself as discussed in Section 3.1.3. Future work could be to re-perform the experiment with other *dr* values between 1.0 and 0.05 and to specifically look at the boundary overlap error associated with them.

It is also surprising that the small arcs method correlated better and had a smaller average distance and standard deviation than the small squares method. While we initially expected them both to perform the same, upon reflection we believe we understand why the small arcs method works better. Unlike the small squares method where the area *dxdx* is constant, the area *drdθ* changes and gets larger as *r* increases since *dθ* remains constant. To illustrate, in Figure 25, the area defined by the arc $dr_1d\theta$ is smaller than the area defined by the arc $dr_2d\theta$. This means overlap errors are smaller near the center and larger near the edges. While this may not make much of a difference while integrating over a geometric area, in this integration the area *drdθ* is multiplied by the probability of the target being in the area which makes all the difference. Near the center, the probability is much higher and tapers off to nearly nothing at the edges. Thus, the small arcs method is better because it is more accurate near the center where it has a greater impact than near the edges where there is little impact. The most pronounced example of the arc method performing better than the squares method is the overlapping region between ellipse 4 and 5. Figure 26 shows this overlapping region (contained in the black rectangle) along with the overlapping region between ellipse 3 and 5. The observation that the boundary goes through the center led us to the conclusion regarding *drdθ*.

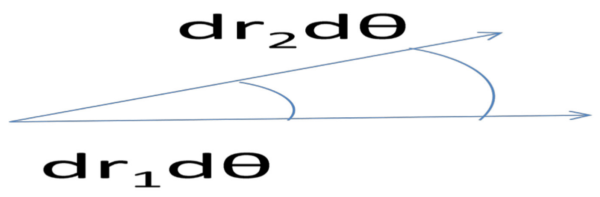


Figure 25 - Illustration of increasing area as r increases.

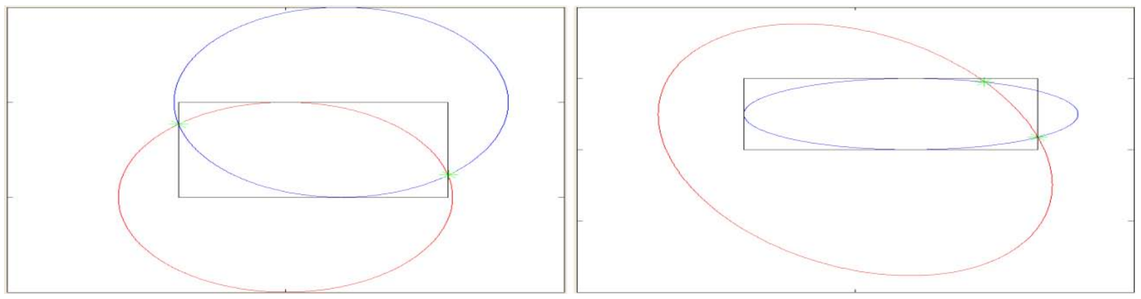


Figure 26 - Visualization of ellipses 3 and 5 (left) and 4 and 5 (right).

It would be natural to conclude that since small arcs is superior to small squares for computing the joint probability two targets are in an overlapping region, it should likewise be used for PPAE. Figure 25 also helps to explain why this path was not pursued. Although the probability a target may be in an arbitrary area can be computed, a convolution integral with varying areas presents numeric computation and programming challenges.

6.1.4 Comparison of Approximation of PPAE, Numeric Calculation of PPAE, and Joint Probability that Two Targets Overlap

PPAE is numerically calculated by convolution integral with *dx* = 0.5 and *dx* = 0.25. PPAE is also approximated by random sampling 10,000 pairs of measurements. Joint probabilities from above are repeated for convenience and as a reference for comparison.

Table 8 - PPAE Computations *dx* = 0.5

Ellipse #s	PPAE by Convolution	PPAE by Random	Joint Probability by Random
1-2	0.025706	0.0197	.5598*.3250 = .1819
1-3	0.17871	0.147	.7624*.7579=.5778
1-4	0.16948	0.1324	.9894*.4561=.4513
1-5	0.068999	0.0636	.4399*.7052=.3102
2-3	0.083835	0.0873	.7659*.7278=.5574
2-4	0.039035	0.0491	.89728.3245=.2911
2-5	0.011254	0.0145	.4516*.2737=.1236
3-4	0.17774	0.1532	.9887*.5031=.4974
3-5	0.011254	0.0156	.4623*.3385=.1565
4-5	0.062818	0.0591	.2701*.9561=.2582

Table 9 - Joint Probability Computations *dx* = 0.25

Equation #s	PPAE by Convolution	PPAE by Random	Joint Probability by Random
1-2	0.020992	0.0197	.5598*.3250=.1819
1-3	0.15601	0.147	.7624*.7579=.5778
1-4	0.15156	0.1324	.9894*.4561=.4513
1-5	0.057569	0.0636	.4399*.7052=.3102
2-3	0.075447	0.0873	.7659*.7278=.5574
2-4	0.03539	0.0491	.8973*.3245=.2911
2-5	0.007908	0.0145	.4516*.2737=.1236
3-4	0.16562	0.1532	.9887*.5031=.4974
3-5	0.007908	0.0156	.4623*.3385=.1565
4-5	0.057872	0.0591	.2701*.9561=.2582

The numeric calculation of PPAE from Equation (51) with *dx* = 0.5 and *dx* = 0.25 is compared to PPAE by random sampling by looking at the correlation coefficients between them. Additionally, the joint probability two targets overlap by random sampling is compared to PPAE by random sampling by looking at the correlation coefficients between them.

Table 10 - Correlation Coefficients for PPAE and Joint Probability Compared to Random Sampling

	Correlation Coefficient
PPAE ; dx = 0.5	0.990896
PPAE ; dx = 0.25	0.992527
Joint Probability	0.907300

Again, we can also look at the average difference and standard deviations. This is not applicable for the comparison of joint probability since the data is dissimilar.

Table 11 - Average and Standard Deviation of Difference for PPAE

	Average Difference	Standard Deviation
PPAE ; dx = 0.5	0.012958	0.013019
PPAE; dx = 0.25	0.008899	0.005591

The numeric calculation of PPAE correlated very well with the random sampling method with both *dx* = 0.5 and *dx* = 0.25 having correlation coefficients greater than 0.99 and very small average difference and standard deviations. As expected, the numeric calculation of PPAE with *dx* = 0.25 outperformed the numeric calculation of PPAE with *dx* = 0.5, however the improvement is modest as indicated by a difference in correlation coefficient of 0.001631. This improvement may not be worth the sixteen-fold computational increase.

Although the joint probability that two targets overlap is different from PPAE, it correlates well with PPAE with a correlation coefficient of 0.9073. Certainly PPAE is more accurate, but what is the effect of using joint probability instead of PPAE? This is very relevant since the joint probability can be approximated quickly with a single rectangle nearly the size of the area of overlap. The use of joint probability instead of PPAE is computationally more efficient. Since the numeric calculation of PPAE can be accomplished in real-time, the use of joint probability instead was not pursued. If improved computational efficiency is needed, joint probability should be explored as an alternative.

6.1.5 Comparison of Numeric Calculation of PPAE vs. Actual Results in a Tracker

PPAE as calculated internally by the tracker is compared to actual AE results of the tracker, thus comparing the prediction to the realization. The Bin column in Table 12 groups PPAE predictions that are numerically near each other and fall within the range designated by the Bin. The Numerator is the total number of association errors related to the Bin. The Denominator is the total number contained in the Bin. The Percentage is the actual association error experienced by the tracker which is determined by dividing the Numerator by the Denominator. The r values (5 and 9) represent the model measurement noise used by the tracker to scale the measurement noise covariance *R* from Equation (9). This provides a representation of association error with relatively small and large sensor error.

Table 12 - PPAE binned vs. actual results

Bin	r = 5			r = 9		
	Numerator	Denominator	Percentage	Numerator	Denominator	Percentage
0%	41138	1203939	3.4170%	7287	1006437	0.7240%
>0% - 5%	23087	61293	37.6666%	3347	58740	5.6980%
5% - 15%	47514	108013	43.9891%	15580	97458	15.9864%
15% - 25%	48786	96932	50.3301%	36810	115245	31.9406%
25% - 35%	43485	77140	56.3715%	39937	96093	41.5608%
35% - 45%	35020	56902	61.5444%	25293	54608	46.3174%
45% - 55%	30919	47305	65.3610%	27546	52285	52.6843%

55% - 65%	25861	38164	67.7628%	33994	59381	57.2473%
65% - 75%	19372	27817	69.6409%	31581	51901	60.8485%
75% - 85%	14545	20135	72.2374%	23440	36879	63.5592%
> 85%	10789	14469	74.5663%	20046	30153	66.4809%
Sum	340516	1752109		264861	1659180	
Average	30956	159282.636	19.4346%	24078.27	150834.5455	15.9634%

The actual association error results obtained with r=5 and r=9 for each bin are compared to the center of the bin (0%, 2.5%, 10%, 20%, ..., 80%, 90%). The correlation coefficients are reported.

Table 13 - Correlation Coefficients for PPAE vs. Actual Results in a Tracker

	Correlation Coefficient
r = 5	0.856047
r = 9	0.955867

The numeric calculation of PPAE agrees well with the actual results in a tracker as indicated by the high correlation coefficients of 0.856 with r = 5 and 0.956 with r = 9. Although the values with r = 5 correlates well, a visual inspection of the numbers shows the predicted value to be far from the actual outcome. For example, when PPAE predicted an association error of between 15% and 25%, the actual association error was about 50%. A visual inspection of the values with r = 9 shows the predicted value to be much nearer the actual outcome. In contrast to the r = 5 value, when PPAE predicted an association error of between 15% and 25%, the actual association error was about 31%. In order to understand this phenomenon, we need to look at the changes that happen as r

changes. Given that *r* is a tuning parameter, a change in *r* creates changes in the operation of the algorithm. At every given point in time after the first timeframe, every state estimate will be different and every PPAE calculation will be different. We first propose an assumption we know to be false; assume every PPAE calculation will be the same. While this assumption is false, it may be that every calculation will be close enough to its counterpoint, or even more likely that in aggregate the calculations of PPAE with $r = 5$ have nearly the same distances between them as do the calculations of PPAE with $r = 9$. So, disregarding the affects of tuning, what else is affected by a change in *r*? We must look at the concept of the steady state innovation covariance (Section 2.2.2.3). As *r* increases, the steady state innovation covariance also increases. An increase in the innovation covariance results in an increased PPAE. This means for every calculation of PPAE done with $r = 5$, the same calculation for PPAE with $r = 9$ will have a higher value. This shifts the bins higher and results in a better correlation coefficient.

The tracker was not tuned prior to this experiment but it would be interesting to see how a tuned tracker performs. The purpose of the experiment was to show that PPAE correlates well with actual values and this was indeed shown without a tuned tracker. The theory behind PPAE assumes that the innovation covariance approximates reality. It seems reasonable that a properly tuned tracker results in an innovation covariance that approximates reality. It is possible PPAE will have the highest correlation with actual values when the tracker is well tuned. If so, PPAE as compared to actual values could be used to tune a tracker or to verify that the tracker is properly tuned.

An additional potential benefit of the PPAE calculation is an indirect measure of the accuracy of the innovation covariance. Trackers often display an ellipse representing a 99% certainty of target location. There is, however, no way of verifying the accuracy of this claim. If PPAE agrees with the actual association error, we have indirectly verified the innovation covariance estimate of the tracker to be accurate.

Finally, in Section 3.9 we discuss that PPAE is predictive and proactive while PAE is reactive. The effect of using PAE instead of PPAE should be looked at for future work. This same experiment can be conducted with PAE by using PAE in the utility function instead of PPAE. Since PAE is more easily calculated than PPAE, this alternative may perform adequately for a lower computational cost.

6.2 Tracker Tuning

6.2.1 Monte Carlo Tuning Required for Multi-target Tracking

In order to determine if Monte Carlo experiments are needed for tuning a Multi-target Tracker, we must compare the results of tuning using a single run to the results of tuning using Monte Carlo results. Table 14 shows the median filter tuning values (*q* and *r*) selected from the Pareto front generated while optimizing for the given run.

Table 14 - Distance to Monte Carlo Pareto front

Run	Median *q*	Median *r*	Closest Distance from Monte Carlo Pareto front
1	13.973066	3.183541	0.001332
2	8.580891	4.014601	0.018573

3	7.385585	4.083661	0.022872
4	13.233166	3.524459	0.006631
5	12.429156	3.854463	0.011437
6	13.821101	7.742915	0.028980
7	11.60978	6.573789	0.006146
8	11.39462	4.340328	0.014361
9	10.363385	3.994608	0.018848
10	10.992313	5.460961	0.002298
Monte Carlo	10.254491	5.846632	N/A

The distance of the far right column is the closest Euclidean distance from the Monte Carlo Pareto front obtained by performing Monte Carlo scoring on the given filter tuning value. This allows us to compare a point that is optimal according to a single run to Monte Carlo optimality. In order to do this, we evaluate the filter tuning values from Table 14 against all runs in the Monte Carlo set, as is inherently done for the Monte Carlo Pareto front. This is deemed Monte Carlo scoring.

The worst NSGA-2 individual average run is data set #6 with a distance of 2.9%. Since run #6 produced the worst individual, we observe its known Pareto front alongside the Monte Carlo Pareto front (Figure 27).

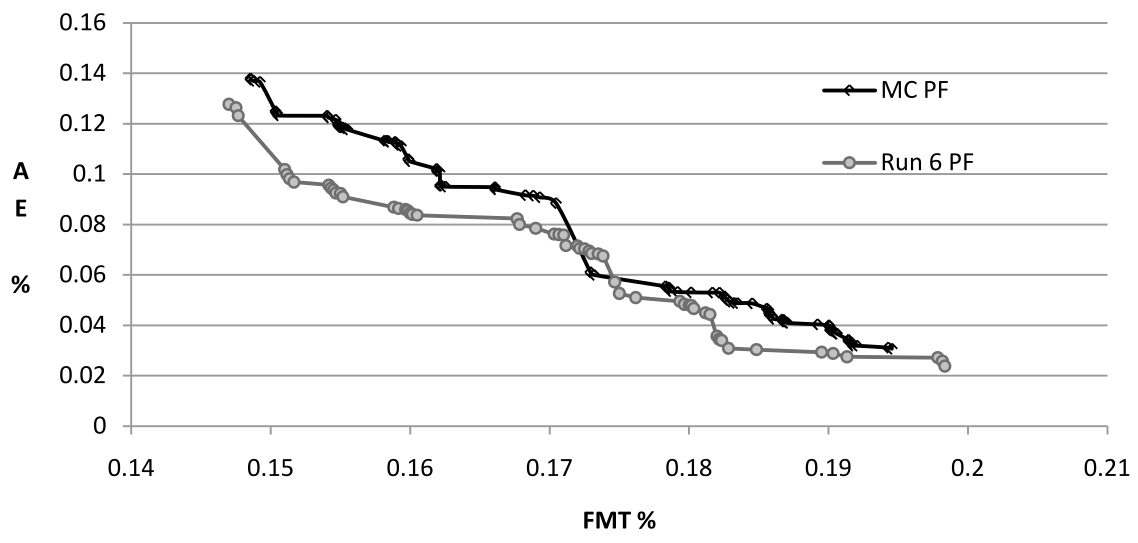


Figure 27 - Comparison of NSGA-2 data set #6 Pareto front and Monte Carlo Pareto front.

Figure 28 shows the Monte Carlo Pareto front along with median selected points after Monte Carlo scoring. It is a graphical representation of the information presented in Table 14.

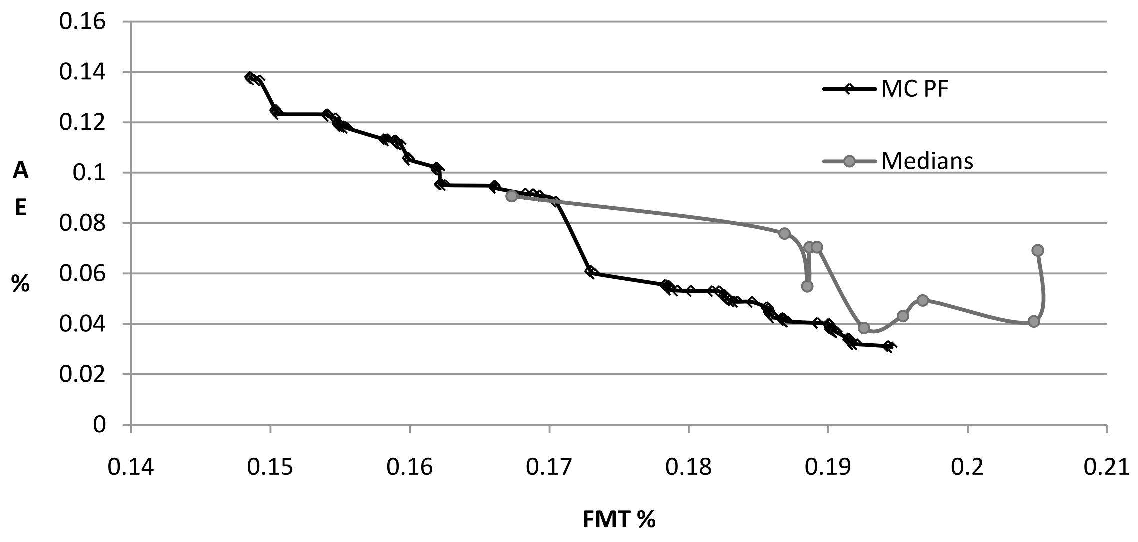


Figure 28 - Comparison of Monte Carlo Pareto front and median selected points after Monte Carlo scoring.

In Figure 27, run #6 seems to largely dominate the Monte Carlo Pareto front. It is deceptive because although the axes for both Pareto fronts are AE% vs. FMT%, the percentages are for either a single run or for all runs. The difference in the way they are evaluated makes it unfair to directly compare the two. Figure 28 highlights this deception by showing where points from Table 14 lie after being evaluated for all runs. In Figure 28, the far right endpoint of the medians series is the farthest point from the Monte Carlo Pareto front, and was generated from Monte Carlo scoring the middle Pareto front point of run #6. In Figure 27 the middle point of run #6 appeared to dominate the Monte Carlo Pareto front, yet Figure 28 shows that it does not dominate after Monte Carlo scoring.

The analysis for this test is comprised of search-space histograms (Figure 29, Figure 30, and Figure 31) and interpolated surface plots (Figure 32, Figure 33, and Figure 34). Figure 29 shows the histogram of the search-space individuals that result in Monte Carlo Pareto front points.

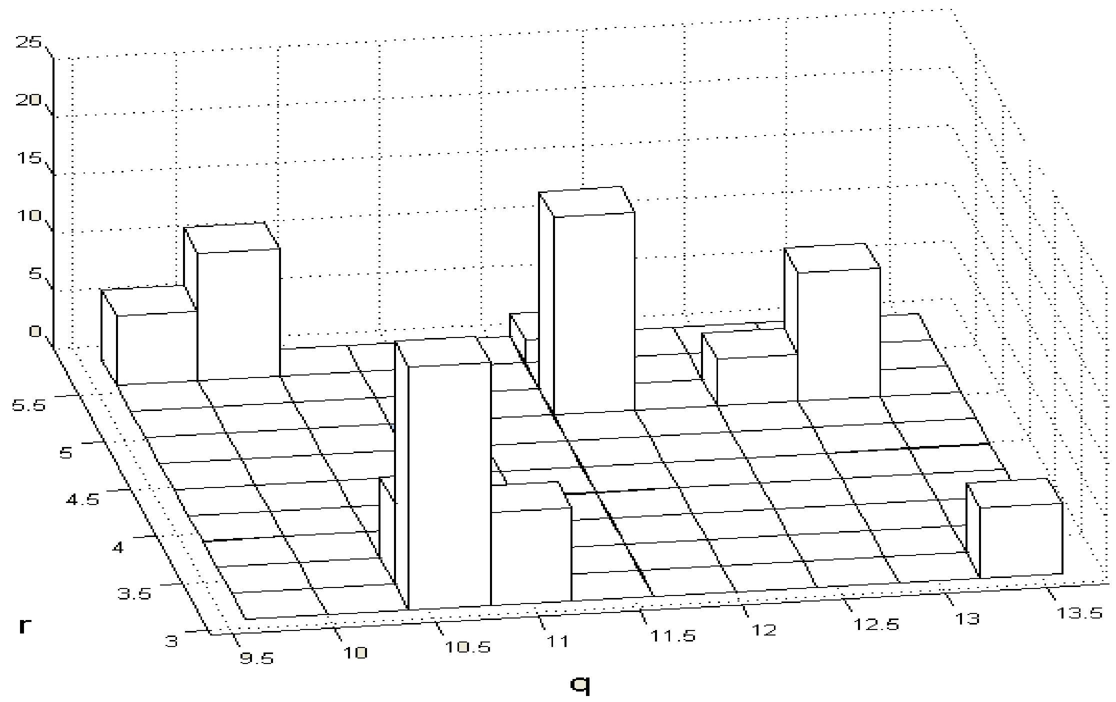


Figure 29 - Histogram of Monte Carlo Pareto front *individuals* over all data sets. This shows the search space (*q* and *r*) which result in Pareto front points.

The *q* vs. *r* tuning parameter histogram of Figure 29 demonstrates the need of examining the search space as well as the objective function space (Pareto front). The strongest statement we can make about the MOEA PFs found in this investigation is that they provide "optimal" solutions for the data being used in the objective functions. While we can and do expect good performance from the tracker using similar tracking data, there is no guarantee such performance would be optimal. With that in mind, we desire to select tuning parameters that are *robust* to changes in data. As shown in Figure 29, the area where *q* varies from 10.5-11.5, and *r* varies from 3-3.5 produces more of the Monte Carlo Pareto front than the area where *q* varies from 13.0-13.5 and *r* varies from 3-3.5. It seems reasonable that the former tuning values may be more robust. Similarly, we look at the *individuals* from run #6 and the combined *individuals* from all 10 runs.

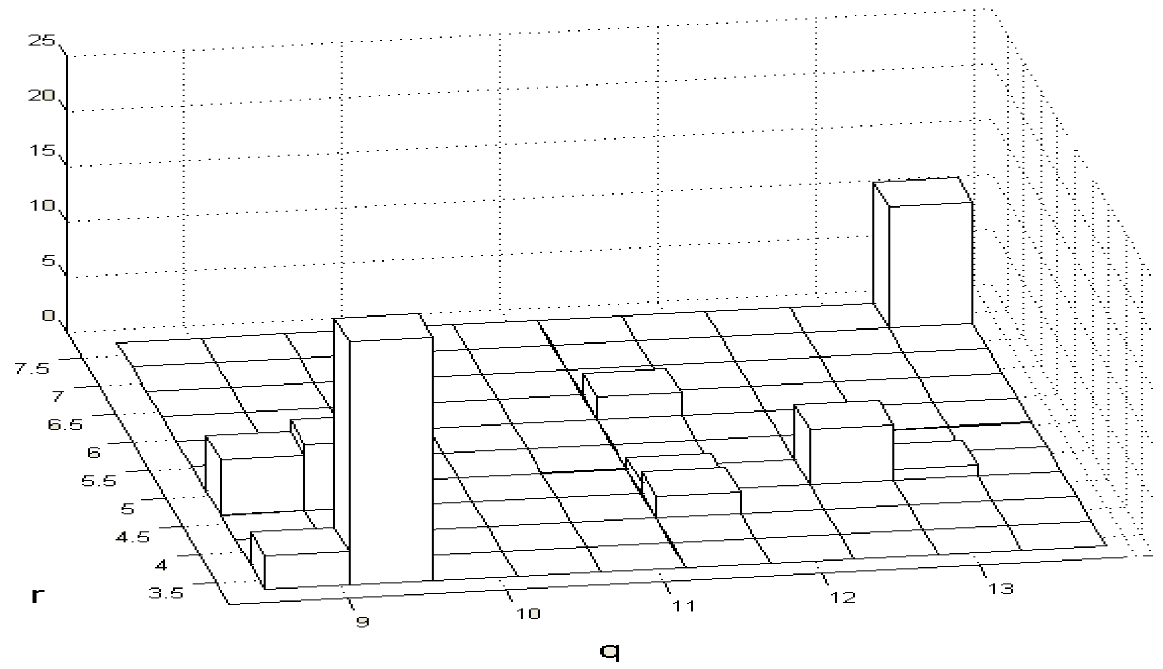


Figure 30 - NSGA-2 Histogram of Scenario # 6 Pareto front *individuals*.

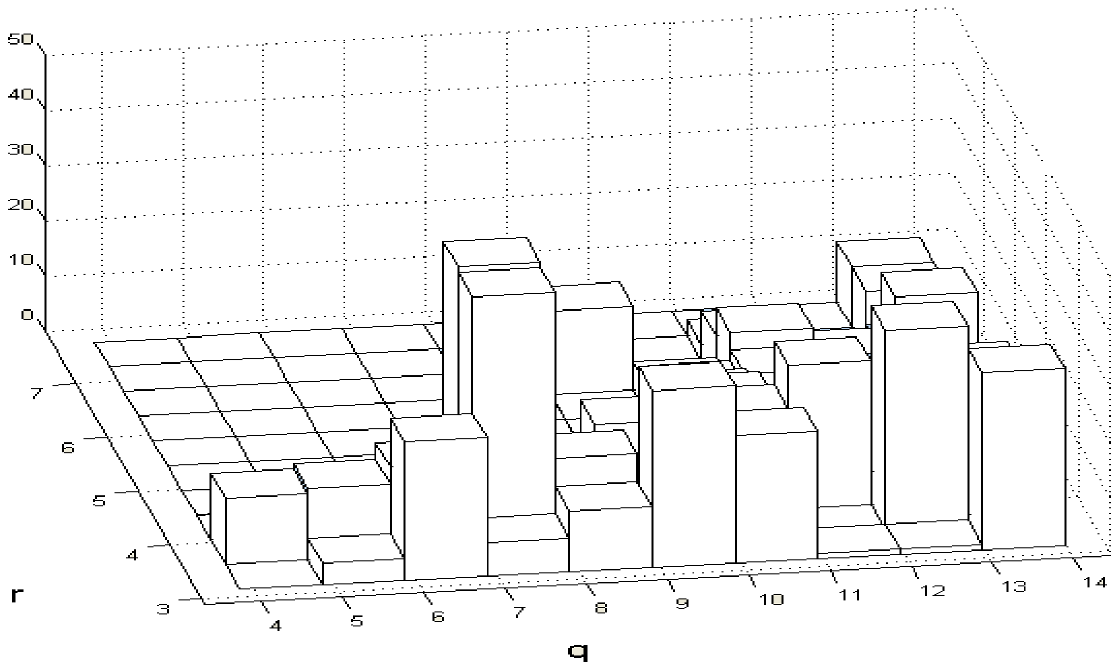


Figure 31 - Histogram of all NSGA-2 individual runs combined Pareto front *individuals* over all 10 data sets for *q* and *r* tuning parameters.

Figure 30 and Figure 31 compare and contrast with Figure 29. Figure 30 shows the individuals which produces the Pareto front for run #6. Figure 31 shows the combined individuals which produces the Pareto front for all individual runs. The disparity between the histograms suggests the results may be more the consequence of the flatness of the evaluation function rather than the similarity of the tuning parameters obtained. Observe that for every individual non-dominated point in Figure 29, there appears to be a corresponding individual non-dominated point in Figure 31 although not in the same amount. This observation does not hold between Figure 29 and Figure 30. This suggests that the set of individuals that produce the Monte Carlo Pareto front (Figure 29) is a subset of the set of individuals that produce all Pareto fronts (Figure 31), but not a subset or superset of the set of individuals that produce a single Pareto front (Figure 30).

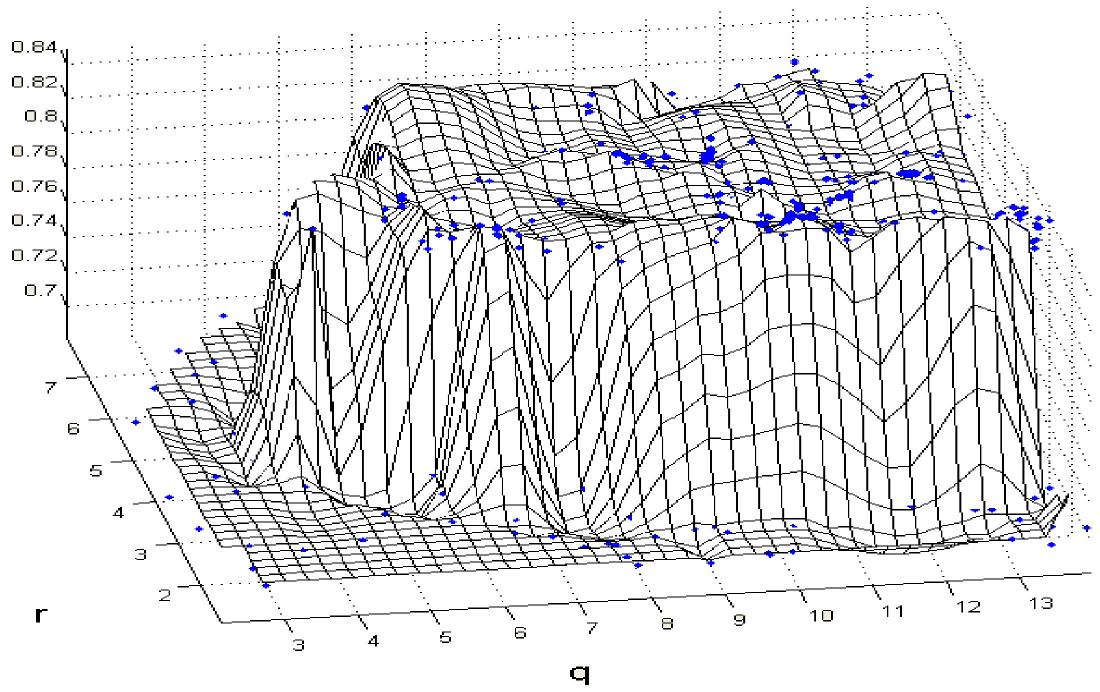


Figure 32 - NSGA-2 interpolated surface plot of 1- association error.

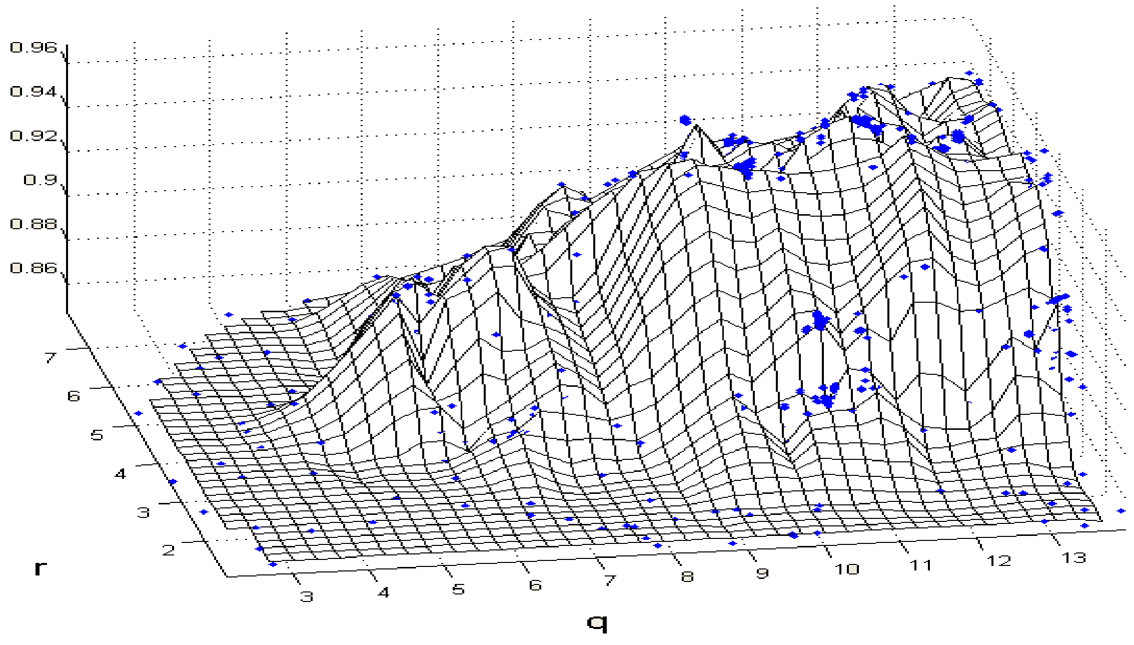


Figure 33 - NSGA-2 interpolated surface plot of 1- FMT.

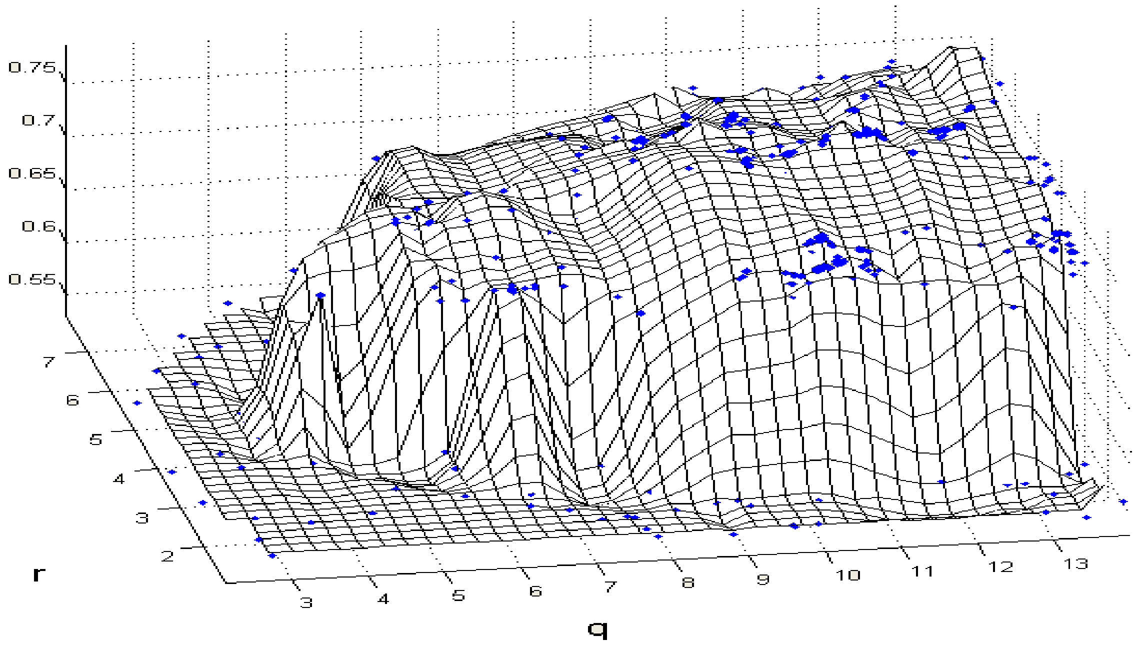


Figure 34 - NSGA-2 interpolated surface plot of 1 – (FMT + association error).

Figure 32, Figure 33, and Figure 34 are all interpolated surface plots using the 9600 evaluations obtained throughout all *generations* while running NSGA-2. A non-interpolated surface plot would have uniformly distributed evaluations. The interpolated surface plots create estimated uniformly distributed data to create the surface plot. For ease of visualization, the desired functions are subtracted from 1 so as to make the desired valleys into peaks. Non-interpolated data is plotted as points. In observing the plots, it seems hard to believe there are 9600 points plotted as many of them are close in value. We see visual evidence of how quickly NSGA-2 converges and that it does not perform a lot of searching even though every *individual* undergoes crossover with a 5% chance of mutation. Figure 32 shows the fitness function for association error. It appears to be a plateau with a certain amount of flatness. Upon closer inspection, we see that the plateau contains gentle rolling hills throughout. This makes sense as we don't expect association error to correlate with tuning values, nor do we expect drastic changes with minor changes in tuning values as would be seen in a more chaotic landscape. Figure 33 shows the fitness function for FMT (or completeness). It appears to be a mountainous slope that increases as both q and r increase. This makes physical sense. Larger values of q and r result in larger association gates, thus the less likely we are to have a vehicle produce measurements outside the gate. Since we continue to track more vehicles without losing them, completeness goes up. Figure 34 shows the search space if FMT and association error are equally weighted. Observe that the tuning values for the Monte Carlo algorithm over all the data sets and the data set #6 NSGA-2 approach generate tuning values on different regions of the fitness landscape plateau. This indicates that the landscapes are different, although we do expect them to have similar properties.

Although the histograms are interesting, the primary question looked at by this experiment is if Monte Carlo is required for tuning MTT. A single experiment such as this cannot definitively answer this question. It can only provide a data point toward making an informed decision. Per Table 14, the average distance of a median point selected from NSGA-2 individual runs and then scored from Monte Carlo analysis is 0.013149 (or 1.3%) with a standard deviation of 0.009191. The worst performer was run #6 with a distance of 2.9%. If we were willing to accept up to a 2.9% reduction in performance, then Monte Carlo would not be needed. Unfortunately, the only way to know this information is to perform the analysis with Monte Carlo which defeats the point of avoiding using Monte Carlo. This experiment needs to be repeated with other data sets to see if these results are consistent and to allow others to make an informed decision regarding the need for continued Monte Carlo analysis.

Although the MOEA effectiveness metrics such as attainment sets, error ratio, hyperarea, and epsilon indicators can be used in statistically comparing the NSGA-2 results with the Monte Carlo results for the given scenario data sets[23], our interest is focused on the computational efficiency factor. That is, we are attempting to show that the NSGA-2 parameter tuning for one of the scenario data sets is generally “close” to the extensive Monte Carlo computation overall data sets.

We have demonstrated the means and ability of tuning MTT algorithm parameters using a multi-objective optimization algorithm over specific scenario data sets. This method produces a known Pareto front of possible tuning parameters for the decision maker to select and is superior to the weighted sums of objectives method which is not guaranteed to produce a Pareto front optimal point.

Moreover, although trackers are traditionally tuned using Monte Carlo runs, for our scenario, choosing a single run to tune with and selecting the median Pareto front point results on average 1.3% away and a maximum of 2.9% away from a Pareto front optimal point using the Monte Carlo runs. Although the point thus selected is near the Monte Carlo Pareto front, it is not necessarily near the median Monte Carlo Pareto front point. This is by no means a proof that we should/should not use Monte Carlo runs. Nor is it believed such a proof can be produced. This is just one data point which may be used to validate a decision to use/not use Monte Carlo runs. Since the objective function in this application is computationally time consuming, the elimination of Monte Carlo runs results in an immediate speedup which may be worth the slightly poorer results.

Monte Carlo has been shown to be appropriate for single-target tracking (STT) via previous research. It has also been shown to be needed for MTT when computing individual statistics/metrics for specific targets being tracked. Monte Carlo has been assumed to be needed for MTT in general. While an individual run may artificially improve the metrics for one specific target, it is also likely the same run artificially worsens the metrics for a different target, thus balancing out. It is possible that as the number of targets increases, Monte Carlo becomes less important.

In addition, multiple objectives may also lessen the importance of a Monte Carlo approach. When a single objective is artificially improved, it is often at the expense of another objective. In a single objective problem formulation, this would go undetected. In a multi-objective problem formulation, the improvement and worsening may balance out, resulting in a flattened fitness function as seen in Figure 34. As the number and diversity of objectives increases, the reliance on Monte Carlo tuning may decrease. If Monte Carlo can be shown to be not needed or of little benefit for a specific MTT algorithm over a multitude of scenario data sets, the computational savings from using a multi-objective evolutionary optimization should make it more practical (efficient) for tuning and comparison of real-time MTT algorithms.

6.2.2 Tracker Tuning for Spatial Sampling

Prior to spatial sampling, the tracker is tuned using NSGA-2 for a framerate of 1 and 30 fps and a measurement noise of 3 and 10. NSGA-2 tunes on the first minute of the simulation, and then the tracker is run with the tuned parameters for the entire ten minutes of the simulation. The tuning parameters thus selected are contained in Table 15. The expected outcome of tuning is that the lower measurement noise will have better performance than higher measurement noise and that a higher framerate will have better performance than a lower framerate. We met the expectation regarding measurement noise.

Table 15 - Tracker Tuning for Spatial Sampling

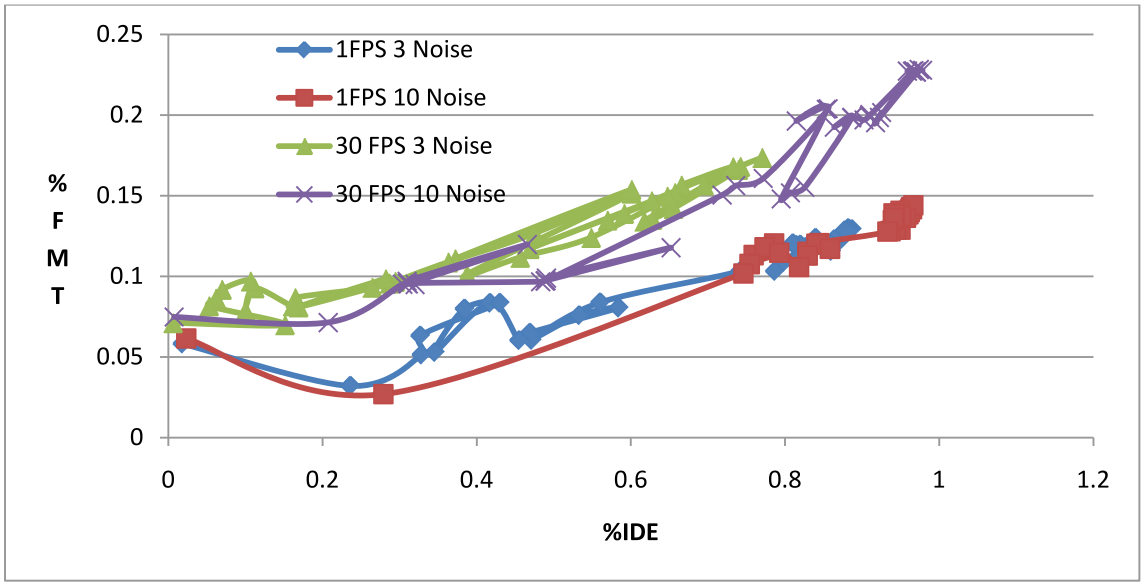
Framerate / Noise	q	r	1 min score association error	1 min score FMT	10 min score association error	10 min score FMT
1 / 3	13.68337	4.493695	0.552923	0.063704	0.8865694	0.1292071
1 / 10	13.818062	6.566797	0.734328	0.096611	0.9647488	0.1416902
30 / 3	0.20765	7.078224	0.401911	0.083318	0.7697684	0.1735797
30 / 10	0.593264	4.243176	0.867203	0.171464	0.9784810	0.2276735

Table 15 shows a measurement noise of 3 outperforms, in both association error and FMT, a measurement noise of 10 for the same framerate at both the 1 minute score and the 10 minute score. The expectation was not met regarding a change in framerate. Table 15 shows a framerate of 30 fps only outperforms a framerate of 1 fps in association error with a measurement noise of 3. Otherwise, a framerate of 1 fps outperforms a framerate of 30 fps. Since this goes against well-established concepts, we must look at the metrics and tracker functioning to understand what is happening. The metrics are all based on confirmed tracks. A track is confirmed when there are 7 out of the last 10 measurements associated with the track. This means a track is confirmed at a minimum initial time of 7 seconds for the 1 fps framerate and 7/30 sec for the 30 fps framerate. Similarly, tracks are deleted when there are 3 or fewer measurements associated with a track in the last 10 measurements. This likewise causes a disparity in deletion time. Because of these differences in the time required to confirm and delete, the metrics are inconsistent with changes in framerate. Any comparison made among factors with the same framerate is valid, but a comparison across framerates is not valid.

6.3 Validation of the Utility Function

A variety of test cases as described in Chapter 5 were used to analyze the utility function construct developed in this research. Table 21 through Table 24 in *Appendix D – Data* show the independent operation of the utility function components (Equal Dispersion, Polling, Missed Measurements, and PPAE), the utility function with equal weightings among the components, and the optimized utility function. A single point from the Pareto front is presented for the optimized utility function in order to compare it with other points in the table. The constant values for the optimized point are given for information only with C1, C2, C3, and C4 representing PPAE, Periodic Polling, Missed Measurements, and Equal Dispersion respectively. The four tables are distinguished by framerate and measurement noise and are thus grouped by all having the same tuning values from Table 15. The variables number of pixels available (10, 100, and 1000) and PCC (70% and 95%) are ordered to show the expected increasing performance. The bounding performances of no HSI and all HSI (or perfect) are also included for comparison. Figure 35 shows a graphical representation of the data presented in *Appendix D – Data*

Figure 35 is the graphical illustration of utility function results. The points were arrayed in the order of their expected performance. If the expectation were entirely met, the curves displayed would be always increasing functions. While the curves can be seen to be mostly increasing, there is some departure that illustrates the variableness in the utility function components. Further analysis is presented that shows the relative performance of the utility function components.



.

Figure 35 - Graphical representation of utility function results.

This analysis is designed to show the performance of the individual components of the utility function along with the equally weighted utility function. The goal of the analysis is to provide a single metric for comparison and generally determine the performance of each component across all variables of interest. The single metric is deemed the *group difference in distance*. The first step toward determining the *group difference in distance* metric is to find the Euclidean distance of (IDE, FMT). This distance is measured from the origin, or from perfection (i.e., (IDE,FMT) = (0,0)). It is generated from the data in Table 21 through Table 24 and the results are tabulated in Table 25 through Table 28. A challenge of having multiple objectives or metrics such as IDE and FMT is determining the relative importance. It is possible that a percent gain in IDE is worth more than a percent loss in FMT; or vice-versa. Euclidean distance provides a fair means of combining the two metrics into a single metric. A liberty is taken with the units for the two metrics. Although the units differ, they are both measured as a percentage and thus scale appropriately when combined by the Euclidean distance.

The next step in determining the *group difference in distance* is to determine the groups. Since the testing conducted is completely matrixed across all variables of interest, data is available for any combination of the variables of interest. A group, therefore, has all the variables of interest constant with the exception of a single variable that is being examined. The single variable we are examining is the component of the utility function; PPAE, periodic polling, equal dispersion, missed measurements, and equal weighting. All other variables of interest (framerate, measurement noise, PCC, and number of HSI pixels available) in a group are constant. For example, the first group from Table 25 has a framerate of 1FPS, a measurement noise of 3, a PCC of 70, and 10 HSI pixels available. It is comprised of the row starting with “Equal Dispersion 10 pix, 70PCC” and ending with the row starting with “Even Weights 10 pix, 70PCC”. The “Optimized” row immediately following each group does not play a role in determining the *group difference in distance* but can be considered an honorary member of the group and is compared to the group after the *group difference in distance* is determined.

Finally, we need to be able to compare across groups. The Euclidean distance cannot be used directly since the values in each group vary widely when compared to other groups. What we really care about is the relative performance within a group. To determine the *group difference in distance* the minimal distance of the grouped components is determined, and the *difference in distance* from the minimum is noted. The *group difference in distance* tells us how much above the best in the group a single component performs, and thus gives us relative performance within the group. A *group difference in distance* of zero indicates the minimal distance of the group, or the best performer. The *group difference in distance* is recorded in Table 25 through Table 28. Now that we have the *group difference in distance* we can compare across all groups. The average and standard deviation of the *group difference in distance* across all variables is found for each component and the equally weighted utility function and the results are tabulated in Table 16. Conclusions from this analysis will be presented after all the analysis is presented.

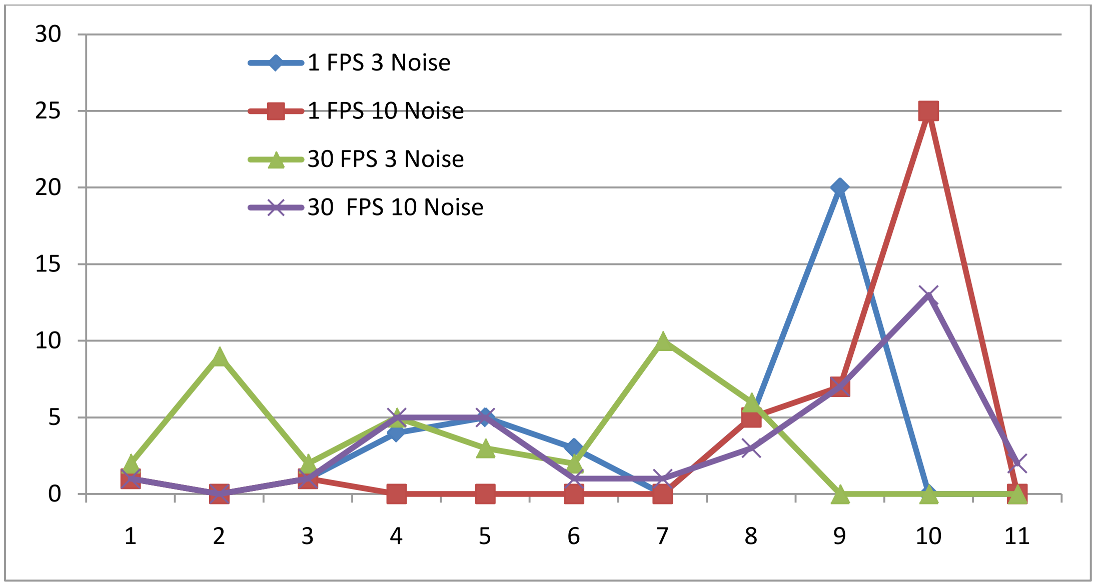


Figure 36 - Histogram of utility component distance to perfection.

Figure 36 is a histogram of the data presented in Table 25 through Table 28 in Appendix D where the higher bin # represents a larger distance from perfection. The histogram of Figure 36 displays vital and meaningful information easily missed by looking only at Figure 35. In Figure 35 the lowermost plot is the case of 1 FPS with a noise of 10. If the lowermost plot was a Pareto front, we would say it dominates and would thus be the superior plot. Figure 36 gives us a better perspective on information contained in Figure 35 which may be too subtle to observe. We see that the majority of points contained in the 1 FPS 10 noise plot are in fact very far away from perfection and thus this plot is in fact the worst performer as expected. In contrast, the 30 FPS 3 noise plot has the majority of points near perfection and is the best performer as expected. This histogram confirms our expectation that a noise of 3 is better than a noise of 10 and a framerate of 30 FPS is better than 1 FPS. Tables 25 through Table 28 provide intermediate analysis used to produce Table 16. Table 16 is the focus of the dissertation and shows the comparative performance of the individual components in the utility function as well as the evenly weighted utility function and the optimized results.

Table 16 - Utility Component Group Distance Averages and Standard Deviations

	Average Grouped Distance	Grouped Distance Standard Deviation
Equal Dispersion	0.066199597	0.065890937
Polling	0.02246603	0.027628111
Missed Measurements	0.061597747	0.06485679
PPAE	0.046002141	0.052415054
Even Weights	0.001017954	0.002071151
Optimized	-0.01431	0.017992

From this table we see that the most natural approach of equally dispersing the resources among the TOIs results in the worst performance and highest average grouped distance of 6.6% and is highlighted in red. Using only missed measurements for dispersing resources is slightly better by .5% (.066 - .061). PPAE used independently is better than equal dispersion by about 2%. Periodic polling has the best individual performance and is better than equal dispersion by about 4.4%. The synergy of the utility function with even weights improves upon periodic polling by about 2.1% and is 6.5% better than equal dispersion. Since the *group difference in distance* measures how much *above* the best performer in a group a component is, the negative value for the optimized utility function represents being *under* or better than the best performer (since "optimized" is not a member of the group) and shows an improvement over even weights by about 1.4%. This represents the maximal additional improvement possible by using optimal weighting in the utility function.

The constant values that resulted in the optimal weighting are reported in Table 21 through Table 24; however there is no consistent resulting optimal weighting. At each optimization, a Pareto front of optimal solutions results and a single value is presented which represents the median weightings, thus making the weighting representative of those for the Pareto front. Moreover, there appears to be no relation between individual performance and weightings. For example, in Table 24 PPAE for 95 PCC and 1000 pixels was the worst individual performer yet had the highest weighting (C1 = 0.79) in the optimal utility function while elsewhere in the same table PPAE for 95 PCC and 100 pixels was the best individual performer and had the lowest weighting (C1 = 0.017). This negative correlation does not hold throughout but is presented to illustrate the lack of correlation.

It is possible to evaluate performance across all the variables and determine a single optimal weighting, although doing so may take months to evaluate even while using a parallel approach as we did. Such a weighting can at most be 1.4% better than even weights. Further, finding a single weighting that works with this data set may not be optimal with other data sets. In short, we deem this approach to be impractical since the time required may not be worth the gain.

Although the test was designed to compare the separate components of the utility function under various parameters (producing Table 16), the same results can be analyzed to compare the individual parameters (producing Table 17 through Table 20) since data is fully matrixed across all variables of interest. The methodology is the same as above, however the groups are changed to compare the desired parameter. In the above analysis, the individual utility function component was varied. In the following analysis, the individual component is fixed (to the evenly weighted utility function; which was generally the best performer in the group) and a different parameter is varied to see how changes in that parameter affect performance. The group distance averages and standard deviations are given to compare PCC, # of pixels, and measurement noise. Once again, a value of zero for the *group difference in distance* represents the best performer in a group and a positive value is the amount above the best performer in the group. Since the average is taken across all groups, an average *group difference in distance* of zero indicates the best performer across all groups, or in all cases.

Tables 17 through Table 20 illustrate the ability to perform tradespace analysis for the new sensor under design. While the number of pixels available will likely be related to the resolution and may not lend itself to tradespace analysis, it is likely that the framerate (which affects PCC and possibly measurement noise) will lend itself to tradespace analysis. A higher framerate will result in less lumens collected each frame and may ultimately reduce PCC. Further, changes in framerate may affect mirror settling time and may impact measurement noise. These effects are unknown at this time, making it impossible to properly model them. Nevertheless, once these effects are known, the downstream consequences of performance can be estimated using the analysis we have provided. This tool can also be used to provide more detailed tradespace analysis as needed.

Table 17 - PCC Group Distance Averages and Standard Deviations

	Average Grouped Distance	Grouped Distance Standard Deviation
PCC = 70	0.057395256	0.05331622
PCC = 95	0	0

Table 17 shows the affect of changing the PCC from 70 to 95. As expected, in all cases a PCC of 95 performs better than that of 70 as indicated by a zero average grouped distance and standard deviation. A PCC of 95 on average was 5.7% better than a PCC of 70.

Table 18 - # Pixels Group Distance Averages and Standard Deviations

	Average Grouped Distance	Grouped Distance Standard Deviation
10 Pixels	0.411736348	0.161534048
100 Pixels	0.298094836	0.106752866
1000 Pixels	0	0

Table 18 shows the affect of changing the number of pixels from 10 to 100 or 1000. As expected, in all cases 1000 pixels performs better than that of 10 or 100 as indicated by a zero average grouped distance and standard deviation. 1000 pixels is on average 29.8% better than 100 pixels and 41.2% better than 10 pixels. 100 pixels is 11.4% better than 10 pixels.

Table 19 - Noise Group Distance Averages and Standard Deviations

	Average Grouped Distance	Grouped Distance Standard Deviation
Noise = 3	0	0
Noise = 10	0.267941728	0.112926047

Table 19 shows the affect of changing the measurement noise. As expected, in all cases a measurement noise of 3 performs better than that of 10 as indicated by a zero average grouped distance and standard deviation. A measurement noise of 3 is on average 26.8% better than a measurement noise of 10.

Table 20 - Framerate Group Distance Averages and Standard Deviations

	Average Grouped Distance	Grouped Distance Standard Deviation
Framerate = 1 fps	0.242461449	0.128361887
Framerate = 30 fps	0	0

Table 20 shows the affect of changing the framerate. As expected, in all cases a framerate of 30 FPS performs better than a framerate of 1 FPS as indicated by a zero average grouped distance and standard deviation. While this is the expected result, it goes counter to the findings of Section 6.9 which suggest the metrics are intransient to changes in framerate. Our data shows the framerate of 30 FPS is on average 24.2% better than a framerate of 1 FPS. Taking into consideration the findings of Section 6.9 we consider this a conservative estimate of the true performance improvement.

To illustrate the use of these tables, let us suppose we face a design decision. The sensor will use 100 pixels, but we can set the framerate either at 1 FPS or 30 FPS. At 1 FPS, we are able to attain a PCC of 95 and a measurement noise of 3. At 30 FPS, we attain a PCC of 70 and a measurement noise of 10. Which should we choose? Since these are parameters we varied in the experiments we could make a direct comparison by looking at the equally weighted entries in Table 21 (0.758, 0.112) for 1 FPS and Table 24 (0.807, 0.152) for 30 FPS and conclude we are better off at 1 FPS by about 6%. Alternately (although less accurately), we could look at Table 20 and see that 1 FPS loses us 0.242, Table 17 and see that 95 PCC gains us 0.057, and at Table 19 and see that a measurement noise of 3 gains us 0.268 to reach the same conclusion that 1 FPS is better by about 8%. By all means, if a direct comparison is possible, use the preceding tables. Tables 24 through 27 give broad guidance about the effect of changing a single variable without the exact knowledge of the other variables to aid in tradeoff decisions.

7 Summary

The contributions of this work are 1) a simulator tool for the creation of simulated vehicle traffic, 2) tuning the multi-target tracker with a multi-objective function, 3) demonstrating that Monte Carlo simulation may not be needed for tuning a multi-target tracker, 4) a tool for the new sensor design tradeoff analysis; analyzing performance space of framerate, probability of correct classification, and number of pixels to be gathered, 5) calculating the probability a target is in an arbitrary region, 6) calculating the joint probability two targets are in the same arbitrary region, 7) PPAE as a useful statistical measure, and most importantly 8) demonstration of the utility function in solving the spatial sampling problem as presented in this dissertation. Each of these contributions is summarized in this chapter.

7.1 Simulator Tool for the Creation of Simulated Vehicle Traffic

Section 4.1 details the simulation tool. Ideally, we would have preferred to use real data with traffic in an urban area and truth data for every vehicle. Although there are many sources for non-truthed data, we were unable to find adequate existing urban data with associated truth. We then looked at the possibility of pseudo-truthing non-truthed data. This technique proposes to use highly effective non-real-time tracking methods along with manual decision-making to produce near-truth from non-truthed data. While this approach should pan out and produce a useable product, at the time the data was needed the technique was too manually-intensive to be of use. The lack of an adequate alternative drove the creation of the simulator tool whose usefulness is illustrated in this dissertation. The merit of the simulation tool is that it produces the effects and challenges a tracker faces while tracking an urban environment while not effecting tracking performance; hence giving a realistic tracking measure. The features the simulation tool provides includes a large number of vehicles, roads, stop signs and traffic lights, differing speed limits and differing vehicle dynamics with respect to the speed limits, vehicle passing, obscuration, parallax, and random generation of many of the vehicle parameters and road curvature. In addition, as discussed in Section 4.8 the tool can be used in a broader means for context-aided experimentation. Without this simulation tool, we could not have produced the tracking results used extensively in the research. When it becomes inexpensive to obtain truth data, or at least simulated truth from real data, then this tool will be obsolete. Until then, there is a tremendous need for simulation tools such as the one developed here.

7.2 Tuning the Multi-target Tracker with a Multi-objective Function

Section 6.9 demonstrates tuning a multi-target tracker with a multi-objective function. Section 2.2.3.5 discusses tracking metrics. Although only three metrics are used in this dissertation, Drummand lists seven tracking metrics [32] and Colgrove lists 15 [56]. We submit that the growth in metrics for multi-target trackers will lead to a growth in using multi-objective functions for tuning and evaluation. Although this work is not the first to perform multi-objective multi-target tracker tuning [57],[58], it is certainly among the first. That being the case, this foundational work highlights the computational challenge of the problem as well as beginning the discussion on the correct metrics to be used.

7.3 Demonstrating Monte Carlo Simulation may not be needed for Tuning a Multi-target Tracker

The greatest impediment to using a multi-target tracker with a multi-objective function is the computational burden. This computational burden is lightened significantly if Monte Carlo simulation is not required. Section 6.8 provides data which may be used to support or reject the use of Monte Carlo simulation. In time, examples may be found to show when it is or is not needed. As more work is done in this area, a more informed decision can be reached. Although this dissertation used Monte Carlo simulation throughout, I believe the same conclusions would have been reached without using Monte Carlo simulation. To definitively say not using Monte Carlo would give the same results would require the analysis be performed with the tracker tuning of a single run (say Run #6 tracker tuning). Since we do not have all of the needed data for a single run, we cannot be certain of the effect of using Monte Carlo simulation other than to say it is an accepted practice and eliminates a source of potential doubt regarding the conclusions.

7.4 A Tool for the New Sensor Design Tradeoff Analysis

Section 6.10 presents an example of how tradeoff analysis can be conducted using the tools provided. The tradespace for the new sensor includes the probability of correct classification (PCC), the number of pixels for which we can obtain HSI data, the measurement noise of the data, and the framerate of the data. Table 17 through Table 20 compares the performance effects of specific values for these parameters. These individual parameters are dependent on the physical design of the sensor and are interdependent among one another; however the exact nature of the interdependency is not fully known. The analysis presented in this dissertation can guide the sensor design by providing performance impact through design decisions. More detailed and specific tradeoff analysis can be performed if needed as more information about the design of the sensor comes to light.

7.5 Calculating the Probability a Target is in an Arbitrary Region

Section 3.1 details the means of calculating the probability a target is in an arbitrary region. Equation (35) provides the means of calculating the probability a target is in a rectangular region and is well-known[54]. Two methods of numerically calculating the probability a target is in an arbitrary region are presented. The first method summates small squares over the arbitrary region and is presented in Equation (34). The second method summates small arcs over the arbitrary region and is presented in Equation (37). Section 6.4 provides validation experiments for both the small squares and small arcs methods. The subsequent results lead to the surprising conclusion that the small arcs method is more accurate.

7.6 Calculating the Joint Probability Two Targets are in the Same Arbitrary Region

One immediate application of being able to determine the probability a target is in an arbitrary region is the ability of determining the joint probability two targets are in the same arbitrary region for the purpose of representing target ambiguity. Equation (38) shows that the joint probability is the product of the individual probabilities. Section 6.4 provides validation experiments for numerically calculating the joint probability by both the small squares and small arcs methods. The results demonstrate accurate calculation of the joint probability.

7.7 PPAE as a Useful Statistical Measure

PPAE may be the singularly most important contribution in this work. The mathematical derivation is presented in Section 3.8 and comprises pages 56 through 63. Initially, the problem is presented and solved in 1-D. The same approach to the problem is applied to the 2-D case, and due to the difficulty in determining a line of equal association error as illustrated by Figure 24 - Visualization of limits of integration for PPAE, we reach a mathematical dead-end in Equation (48). Since it is possible to determine for specific measurements if an association error will occur, a numeric approach is used and results in Equation (51). This numeric solution is validated by comparing it to a random sampling estimated solution in Section 6.5 with the results showing validation. Even though this shows PPAE can be accurately numerically calculated, its accuracy and usefulness needed to be verified in an actual multi-target tracking environment since the underlying assumptions do not hold. PPAE looks at the interactions between two tracks and the association errors that result as two targets get physically near each other, yet a real tracking environment has multiple vehicles that can cause more complex association errors. Additionally, one of the assumptions of the PPAE is that the covariance of the tracker is in fact representative. Since this is an unverifiable assumption, the performance of PPAE in a real tracker is in question. Section 6.6 presents an experiment whereby numerically calculated values from PPAE are compared to actual resulting association errors in the tracker. The results show that although PPAE may not accurately depict actual association errors (and that it depends upon the tuning of the tracker), PPAE does correlate very well with association errors and is therefore useable in the needed context. The final validation of PPAE occurs in the utility function discussed in the next subsection.

Although PPAE has an immediate application for which it was developed, it may have other as yet unknown applications. The math behind the derivation may be applicable to other problems.

7.8 Solving the Spatial Sampling Problem

The entire point of the dissertation is to solve the spatial sampling problem. The problem is presented formally in Section 3.6 and the proposed solution of using a linear combination of heuristics known as the utility function that is optimized is presented in Section 3.15. In order to have a basis of comparison, each individual component of the utility function or heuristic is tested independently and the performance is compared to using the utility function with even weights and optimized weights. The experiment is detailed in Section 6.10 along with the results. PPAE as part of the utility function receives further validation and is shown to be useful in the experiments for the utility function. PPAE was the 2nd best individual heuristic (periodic poling was best) and contributed to the improved performance of the utility function over periodic poling.

The solution to the spatial sampling problem is the core of a RSM for the new MOS sensor. We successfully showed that this sensor can perform persistent surveillance with highly improved tracking performance over panchromatic video. The simulation and testing showed that for tracking 100 vehicles, having a resource of 1000 pixels approaches the performance of a HSI-only sensor at the same speed. The advantage of the MOS sensor, of course, is that since it does not collect all the HSI data, it can operate at faster speeds and thus can show an improvement over panchromatic video.

8 Bibliography

[1] R. Grant, "The Fallujah Model," *Air Force Magazine*, pp. 48-53, Feb. 2005.

[2] S. A. James, *A Small-Scale 3D Imaging Platform for Algorithm Performance Evaluation*, A. F. I. o. Technology, Ed. Dayton, OH, 2007, Thesis.

[3] E. R. Harshberger, "Global Implications for the U.S. Air Force," *Rand Review*, 2002.

[4] L. W. Kang and C. S. Lu, "Pwwer-rate-distortion Optimized Resource Allocation for Low-complexity Multiview Distributed Video Coding," Institute of Information Science, Academia Sinica, IIS Techinicla Report TR-IIS-08-003, 2008.

[5] T. Yeh, Y. Chen, and Z. Liao, "An Image Stitching Process Using Band-Type Optimal Partition Method," *Asian Journal of Information Technology*, vol. 7, no. 11, pp. 498-509, 2008.

[6] P. W. Boettcher and G. A. Shaw, "Energy-constrained Collaborative Processing for Target Detection, Tracking and Geolocation," in *Proc. of IPSN*, 2003, pp. 254-268.

[7] Y. Bar-Shalon and X. Li, *Multi-target-Multisensor Tracking: Principles and Techniques*. Danvers, MA: Copyright Clearance Center, 1995.

[8] J. Kerekes, M. Muldowney, K. Strackerjan, L. Smith, and B. Leahy, "Vehicle Tracking with Multi-temporal Hyperspectral Imagery," in *SPIE Proceedings, Algorithms and Technologies for Multispectral,Hyperspectral, and Ultrapsectral Imagery XII*, vol. 6233, Orlando, FL, 2006.

[9] P. Arambel, M. Antone, C. Rago, H. Landau, and T. Strat, "A Multiple-Hypothesis

Tracking of Multiple Ground Targets from Aerial Video with Dynamic Sensor Control," in *Proc. of SPIE 5429*, 2004, pp. 23-32.

[10] N. A. Soliman, *Hyperspectral-Augmented Target Tracking*. Dayton: Air Force Institute of Technology, 2008, Thesis.

[11] T. Wang and Z. Zhu, "InIntelligent Multimodal and Hyperspectral Sensing for Real-Time Moving Target Tracking," in *Proceedings of the 2008 37th IEEE Applied Imagery Pattern Recognition Workshop* , 2008, pp. 1-8.

[12] N. Steinberg, C. L. Bowman, and F. E. White, "Revision to the JDL Data Fusion Model," in *Joint NATA/IRIS Conference*, Quebec City, 1998.

[13] R. B. Washburn, M. K. Schneider, and J. J. Fox, "Stochastic Dynamic Programming Based Approaches to Sensor Resource Management," in *Proceedings of the Fifth International Conference on Information Fusion*, 2002, pp. 608-615.

[14] M. K. Schneider, A. Nedich, D. Castanon, and B. Washburn, "Approximation Methods for Markov Decision Problems in Sensor Management," DARPA Integrated and Sensing Processing Program AFRL-SN-WP-TR-2005-1108, 2006.

[15] P. Ross, "Hyper-Heuristics," in *Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Techniques*, E. K. Burke and G. Kendall, Eds. Springer, 2005, ch. 17, pp. 529-556.

[16] E. Hart and P. Ross, "A Heuristic Combination Method for Solving Job-Shop Scheduling Problems," in *Parallel Problem Solving from Nature*, A. E. Eiben, et al., Eds. Springer-Verlag, 1998, pp. 845-854.

[17] L. Varsano and S. Rotman, "Point Target Tracking in Hyperspectral Images," in *Proceedings of the SPIE*, vol. 5806, Orlando, FL, 2005, pp. 503-510.

[18] SIBR Proposal. Network-Centric Urban Vigilance. [Online]. http://www.dodsbir.net/SITIS/archives_display_topic.asp?Bookmark=29685

[19] T. Cooley. Validation and Experiment Plans for a Space Based Hyperspectral Imaging System. [Online]. http://ieeexplore.ieee.org/iel5/6913/18598/00857214.pdf

[20] J. Yang and V. Honavar, "Feature Subset Selection Using a Genetic Algorithm," in *IEEE Intelligent Systems*, 1998.

[21] W. Siedlecki and J. Sklansky, "A Note on Genetic Algorithms for Large-Scale Feature Selection," in *Pattern Recognition Letter*, 1989, p. 335–347.

[22] X. Li and V. Jilkov, "A Survey of Maneuvering Target Tracking: Dynamic Models," in *SPIE Proceedings. Signal Data Processing. Small Targets*, Orlando, FL.

[23] P. Maybeck, *Stochastic Models, Estimation, and Control, I*. New York, NY: Academic Press, Inc., 1979, Republished: Arlington, VA: Navtech, 1994.

[24] A. Lipton, H. Fujihoshi, and R. Patil, "Moving Target Classification and Tracking from Real-time Video," in *IEEE Workshop on Applications of Computer Vision, WACV '98,* Princeton, NJ, 1998, pp. 8-14.

[25] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings DARPA Image Understanding Workshop*, 1981, pp. 121-130.

[26] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*.

Norwood, MA: Artech House, 1999.

[27] Y. Bar-Shalon and T. Fortmann, "Tracking and Data Association," *Mathematics in Science and Engineering*, vol. 179, 1988.

[28] Y. Bar-Shalon, T. Kirubarajan, and C. Gokberk, "Tracking with Classification-aided Multiframe Data Association," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, 2005, pp. 868-878.

[29] F. Dellaert, *Monte Carlo EM for Data-Association and its Application in Computer Vision*. Carnegie Mellon University, 2001, PhD thesis.

[30] F. Castella, "Sliding Window Detection Probabilities," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 12, 1976, pp. 815-819.

[31] S. Blackman, *Multiple Target Tracking with Radar Applications*. Norwood, MA: Artech House, 1986.

[32] R. Rothrock and O. Drummond, "Performance Metrics for Multiple-Sensor, Multiple-Target Tracking," in *SPIE Proceedings on Signal and Data Processing of Small Targets*, vol. 4048, Orlando, FL, 2000, p. 521–531.

[33] S. Colegrove, B. Cheung, and S. Davey, "Tracking System Performance Assessment," in *Proceedings of the Sixth International Conference of Information Fusion*, vol. 2, Cairns, Queensland Australia, 2003, p. 926–933.

[34] K. Tarplee, *Technical Discussions/Interview*. Jan. 28, 2008, Numerica Corporation.

[35] Wolfram. Ellipse. [Online]. http://mathworld.wolfram.com/Ellipse.html

[36] Wolfram. Rotation Matrix. [Online].

http://mathworld.wolfram.com/RotationMatrix.html

[37] B. Secrest. Ellipse. [Online]. Correction to http://mathworld.wolfram.com/Ellipse.html

[38] D. Watkins, *Fundamentals of Matrix Computations*. New York, NY: Wiley, 1991.

[39] M. Michalewicz and D. Fogel, *How to Solve it*. Berlin, Germany: Springer, 2000.

[40] T. Bäck, D. Fogel, and Z. Michalew, Eds., *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

[41] C. Coello, D. Van Veldhuizen, and G. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York, NY: Kluwer Academic Publishers, 2002.

[42] M. Kleeman, *A Comparison of NSGA-II and SPEA-2*. Publication Pending.

[43] D. Van Veldhuizen and G. Lamont, "Evolutionary Computation and Convergence to a Pareto Front," *Late Breaking Papers at the Genetic Programming Conference*, pp. 221-228, 1998.

[44] MicroImages, Inc. Introduction to Hyperspectral Imaging. [Online]. http://www.microimages.com/getstart/pdf/hyprspec.pdf

[45] J. Blackburn, et al., "Feature Aided Tracking with Hyperspectral Imagery," in *SPIE Optics and Photonics*, San Diego, CA, 2007.

[46] C. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. New York, NY: Kluwer/Plenum, 2003.

[47] J. Benediktsson and I. Kanellopoulos, "Classification of multisource and hyperspectral data based on decision fusion," *IEEE Transactions on Geoscience and*

Remote Sensing, vol. 37, no. 3, pp. 1367-1377, May 1999.

[48] B. Hammer and T. Villmann, "Generalized Relevance Learning Vector Quantization," *Neural Networks*, vol. 15, no. 8, p. 1059–1068, 2002.

[49] K. Oehler and R. Gray, "Combining Image Compression and Classification Using Vector Quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, p. 461–473, 1995.

[50] B. Hammer, M. Strickert, and T. Villmann, "Learning Vector Quantization for Multimodal Data," *ICANN'02*, pp. 370-375, 2002.

[51] S. Schweizer and J. Moura, "Efficient Detection in Hyperspectral Imagery," *IEEE Transactions on Image Processing*, vol. 10, p. 584–597, Apr. 2001.

[52] Z. Zhang, N. Younan, and C. O'Hara, "Wavelet Domain Statistical Hyperspectral Soil Texture Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 3, p. 615–618, 2005.

[53] C. Chang and H. Ren, "An Experiment-based Quantitative and Comparative Analysis of Target Detection and Image Classification Algorithms for Hyperspectral Imagery," *IEEE Trans. Geosci. Remote Sensing*, vol. 38, p. 1044–1063, Mar. 2000.

[54] Wolfram. Distribution Function. [Online]. http://mathworld.wolfram.com/DistributionFunction.html

[55] B. Secrest, *Traveling Salesman Problem for Surveillance Mission Using Particle Swarm Optimization*. Dayton, OH: School of Engineering and Management of the Air Force Institute of Technology, Air University, 2001, Thesis.

[56] S. Colegrove, B. Cheung, and S. Davey, "Tracking System Performance Assessment," in *Proceedings of the Sixth International Conference of Information Fusion*, vol. 2, 2003, p. 926–933.

[57] J. García, et al., "Application of Evolution Strategies to the Design of Tracking Filters with a Large Number of Specifications," in *Applied Signal Processing*, vol. 8, 2003, p. 766–779.

[58] J. Besada, et al., "Design of IMM Filter for Radar Tracking Using Evolution Strategies," in *IEEE Transactions on Aerospace and Electronic Systems*, [serial online], 2005, pp. 1109-1122.

[59] J. Kerekes, Z. Ninkov, and A. Raisanen, *Technical Discussions/Interview*. Aug. 17-24, 2007, Faculty of Rochester Institute of Technology.

9 Appendix A – Tunable Spectral Polarimeter

Since this research makes use of a new sensor under development, we will describe the tunable spectral polarimeter (TSP) and the multi-object spectrometer (MOS). These sensors have the capability of being combined with the techniques described in this dissertation to provide robust target tracking.

9.1 Tunable Spectral Polarimeter

The TSP extends the concept currently used in visible color cameras to multi-modality sensing by grouping the modalities in 2x2 arrays integrated onto a detector focal plane. Figure 37 illustrates this concept for polarization sensing together with a Fabry-Perot etalon spectrometer where all modes are sensing the same spatial location. Each spatial pixel (thick lines on array) is actually a 2x2 array of detectors with each detector having an integrated filter to sense polarization or spectral information. With the three polarization channels, this concept can measure the S_0, S_1, and S_2 Stokes parameters which characterize the intensity, degree and angle of linear polarization. Given the irradiance E_α falling on each polarized detector at angle α, the Stokes parameters are calculated as [59]

$$\begin{bmatrix} S_0 \\ S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} \frac{2}{3}(E_0 + E_{60} + E_{120}) \\ \frac{2}{3}(2E_0 - E_{60} - E_{120}) \\ \frac{-2}{\sqrt{3}}(E_{120} - E_{60}) \end{bmatrix} \tag{56}$$

with the resulting degree of polarization (*DOP*) and angle of polarization (*AOP*) computed as [59]

$$DOP = \frac{\sqrt{S_1^2 + S_2^2}}{S_0} \tag{57}$$

$$AOP = \frac{1}{2}\tan^{-1}\left(\frac{S_2}{S_1}\right) \tag{58}$$

Note that the intensity of a pixel can be recovered as the S_0 parameter. These polarization measurements along with the spectral information provide features useable for the detection and feature aided tracking of desired targets.

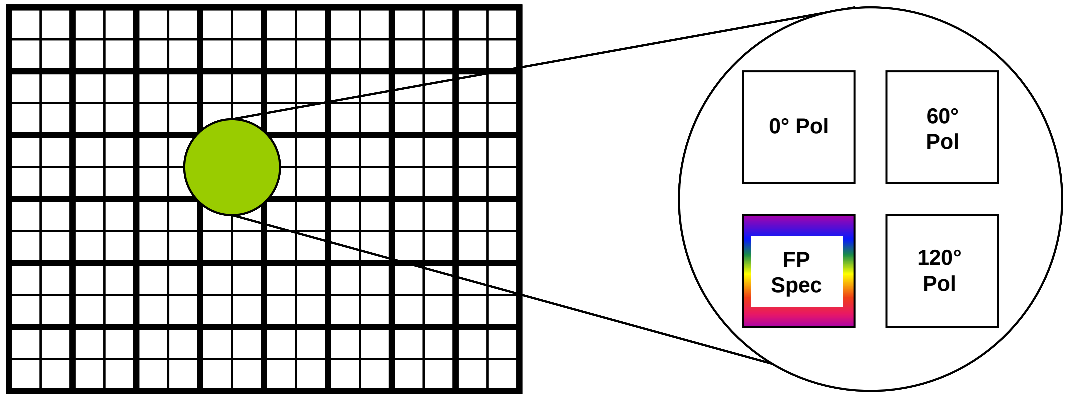


Figure 37 - Illustration of the tunable spectral polarimeter.

9.2 Multiobject Tracking Spectrometer

The idea of combining an array of MEMS micromirrors with an imager and a spectrometer array is the concept behind a revolutionary new implementation of standard multi-object spectrometer (MOS) that is being explored by researchers[59]. Figure 38 illustrates the basic operation of the new concept. Each mirror in the array can be moved individually to reflect light into one of two directions. The micromirror array is located at an intermediate focal plane of the optical system. If all the individual mirrors are turned in the same direction, then an image is formed on a focal plane array (e.g., a visible wavelength CCD producing RGB video). The video can be analyzed in real time using existing technology and appropriate targets identified. Mirrors co-incident with targets of interest can then be tipped and the associated light will pass through a spectrometer and be recorded as a spectrum on a second focal plane array. The resulting compound image is illustrated in Figure 39.

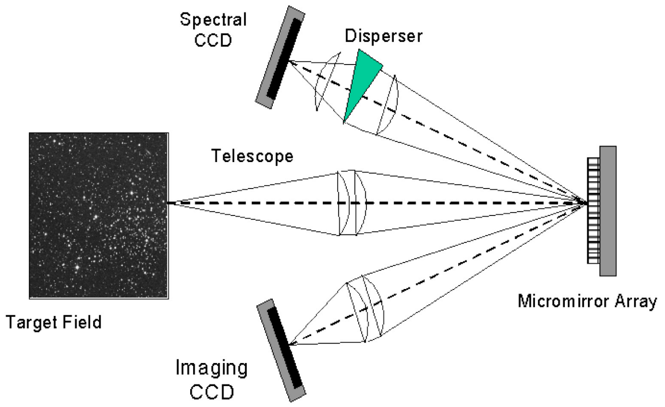


Figure 38 - Illustration of micromirrors.

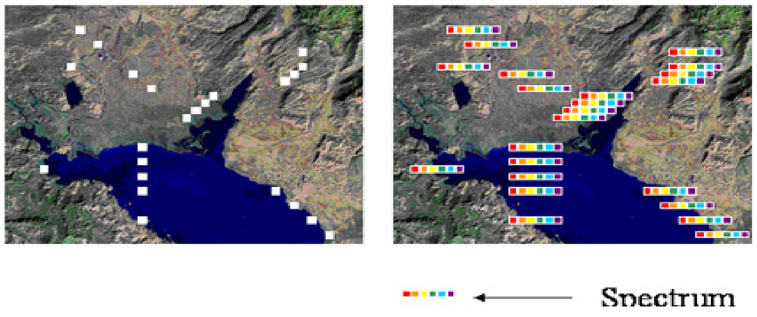


Figure 39 - Compound EO/HSI image. White pixels in the left image are those for which HSI data is collected (hence no EO data). The right image illustrates the greater information of HSI data collected at strategic pixels of the image.

9.3 Factors to Consider

We wish to maximize framerate and probability of correct HSI classification (PCC). These two objectives naturally oppose each other. The framerate depends on the sensor dwell time, which affects the signal to noise ratio (SNR) quality of the HSI data. If we increase the framerate by reducing the sensor dwell time, the SNR of the HSI data decreases, making it more difficult to correctly classify targets and decreasing the probability of correct HSI classification.

For practical reasons, we wish to use a constant framerate and a constant and aligned spectral band size. Although it is possible to use time-varying framerates which produce gains in performance, existing software usually does not do so since the added software complexity is not worth the performance gains. Likewise, frequency dependant band sizes would allow a constant framerate for a given SNR (differing wavelengths result in differing SNR, hence we can change the band size rather than dwell time to control SNR), however existing classification routines may need to be modified to match bands which are not correctly aligned, so a constant band size is desired.

10 Appendix B – Feature Subset Selection

10.1 Feature Subset Selection

The primary difference between the MOS and the TSP is that we can control the bands of HSI data to be collected with the TSP. This feature enables us to maintain a higher framerate by reducing the HSI data collected. Once the pixels for which HSI data is to be collected are determined through optimizing the utility function, we then must determine the appropriate bands for each pixel to collect.

Although the dissertation solves spatial sampling which is needed for either type of sensor, this dissertation does not attempt to solve feature subset selection which would be needed for the TSP. An early decision steered us toward the MOS and so feature subset selection was not needed. Some thought, however, was put forward toward a solution to feature subset selection and those thoughts are presented here.

10.2 Optimization of the Feature Subset Selection Problem

We have chosen to formulate the problem in terms of framerate and probability of correct classification (PCC). This is just a personal preference since framerate is directly related to dwell time (dwell time = 1/framerate). Framerate seems more natural since it is a parameter used in the tracker. Likewise PCC seems more natural since it is the desired effect. We desire to solve the following:

$$\begin{aligned}&\textit{Minimize } F = (f1, f2, f3), \textit{where}\\ &f1 = 1/\ \textit{framerate}\\ &f2 = 1 - \text{PCC}\\ &f3 = \text{ \# of Bands Collected}\end{aligned} \tag{59}$$

In this problem, an *individual* consists of the framerate (an integer value of # of frames per second, between 1 and 100), the bandwidth (an integer value between 1 and 100 which is multiplied by 0.01 to give the bandwidth), and the specific bands to collect (see below for a description). The *population* is a collection (of size 2000) of *individuals*. A *generation* consists of: 1) Producing a random sampling of HSI pixels to be tested 2) Receiving simulated data dependant on the dwell time and band size of only those bands indicated by the individual. 3) Mixing pixels in simulation to further corrupt data and account for background mixing and partial pixels. 4) Feeding all the pixels to the classifier. 5) Comparing the results of the classifier with the truth to calculate PCC. Since f_1 and f_3 are inherent in the individual, we can score the individuals and reiterate. The Pareto front will allow us to make an informed choice of framerate needed to achieve an acceptable PCC for a given # of bands collected. The associated individual tells us the specific bands to collect.

Note: Using existing data collected with a 0.01 micron band size, we have 2500 bands. Water vapor vastly increases SNR at 1.4 and 1.9 microns, making affected frequencies unusable. Removing them, we are left with 1950 bands. These can then be down-sampled to simulate any band size of x * 0.01. For example, if we want a band size of 0.10, we multiply 0.01 by 10 to get 0.10, thus we now have 195 bands. The *individual* contains 1950 bits representing the original 1950 bands of .01 band size. A dilation operator will be included to fill bands with the appropriate bits. If a single bit in the range of the band size is a 1, all the bits within the band size become a 1. Suppose a 0.01 band size individual is mated with a 0.1 individual and the result is a 0.3 individual. The result of the mating is that in the first 30 bits of the child, only 1 bit is a 1 (inherited from the .01 parent). The dilation operation would change all of the first 30 bits to a 1 to account for this. Although we feel an opposite operation is needed to turn off bits in a 0.01 child that it inherited from a 0.3 parent, we do not conceive how it would work as of yet. We might randomly only turn on only 1 of the 30 bits, or we might see if the archive 0.01 individual has bits there and only turn those on.

10.3 Data Specific and Real Time Issues of Spectral Sampling

The purpose of collecting the HSI data is to be able to classify the vehicles (see Section 2.2.8). The solution produced using spectral sampling is certainly data specific. It needs to be! The pre-existing classes we are using to classify against determine the specific bands we should collect. How then can this be a useful real-time solution? We propose that as we create the classes (in real-time), we feed them to a process (perhaps on another computer) that proceeds to find the solution we need. As it is changed, the perturbed new set of classes can also be solved, with the expectation that the prior solution will be similar to the new solution and hence the current individual population should swiftly find good solutions.

11 Appendix C - Miscellaneous

11.1 Simulation Source Code

The simulation source code is written in Matlab and includes Sim.m (the main routine), Initialize.m, Initialize_Car.m, Propagate_Cars.m, Obscured.m, Parallax_active.m, Grow_Truth.m, Draw_Image.m, and genRuns.m.

11.2 Search Utility

Another heuristic was considered for inclusion in the utility function. This heuristic which we call the search utility is akin to the periodic polling utility, but looks at the time each pixel was last visited rather than the time for each vehicle. It performs a scan or random search of the area. The data for which the experiments were conducted did not contain a largely obscured area and thus this utility would not have been useful and so was not included. This utility is defined as:

$U_{ij}^{S}(t)$: Search utility which models the utility associated with acquiring new targets. This value may depend on context related information if available. For example, the utility of searching for new targets where a building exists may be low, but where a forest exists, it may be higher. The purpose of this function is to produce a search pattern or scanning over the desired area. An example might be:

$$U_{ij}^{S}(t) = 1 - e^{-\alpha(t-\tau)} \tag{60}$$

where τ is the time of the last HSI sample of the *ij*th pixel and α is a decay rate value to control the growth of the utility function. Thus, every pixel begins with a utility of 0 (when $t = \tau$) which gradually increases toward unity with time of non-sampling. If searching is all that is occurring, this ensures every pixel is searched equally.

12 Appendix D – Data

These tables are removed from the main body of the test to enable continuity while reading. Tables 21 through Table 24 are the results of the test used to validate the utility function detailed in Section 5.13 and discussed in Section 6.10. Tables 25 through Table 28 are analysis of the results detailed and discussed in the same sections.

Table 21 - 1 Frame/Sec ; Noise = 3

1FPS,3 noise, 10 Min	IDE	FMT
No HSI	0.8865694	0.1292071
Equal Dispersion 10 pix, 70PCC	0.8818164	0.1298434
Polling 10 pix, 70PCC	0.8750451	0.1238300
Missed Measurements 10 pix, 70PCC	0.8822744	0.1285421
PPAE 10 pix, 70PCC	0.8668943	0.1211481
Even Weights 10 pix, 70PCC	0.8698220	0.1213586
Optimized 10 pix , 70PCC, C1= 0.261027, C2= 0.166323 , C3 = 0.485348, C4 = 0.087302	0.859824	0.115643
Equal Dispersion 10 pix, 95PCC	0.8727833	0.1230774
Polling 10 pix, 95PCC	0.8736999	0.1264411
Missed Measurements 10 pix, 95PCC	0.8774305	0.1285236
PPAE 10 pix, 95PCC	0.8638946	0.1230084
Even Weights 10 pix, 95PCC	0.8594438	0.1197020
Optimized 10 pix , 95PCC, C1= 0.223091, C2= 0.206052 , C3 = 0.275300, C4 = 0.295557	0.854185	0.117933
Equal Dispersion 100 pix, 70PCC	0.8394446	0.1238333
Polling 100 pix, 70PCC	0.8141199	0.1194663
Missed Measurements 100 pix, 70PCC	0.8326414	0.1193468
PPAE 100 pix, 70PCC	0.8195028	0.1122828
Even Weights 100 pix, 70PCC	0.7982077	0.1086886
Optimized 100 pix , 70PCC, C1= 0.006262 C2= 0.329008 C3 = 0.424716 C4 = 0.240014	0.786730	0.103219

Equal Dispersion100 pix, 95PCC	0.8196162	0.1196347
Polling 100 pix, 95PCC	0.7679606	0.1144815
Missed Measurements 100 pix, 95PCC	0.8100332	0.1204579
PPAE 100 pix, 95PCC	0.7851347	0.1167205
Even Weights 100 pix, 95PCC	0.7580832	0.1119949
Optimized 100 pix , 95PCC, C1= 0.011404 C2 = 0.189839, C3 = 0.452648, C4 = 0.346109	0.741823	0.103286
Equal Dispersion1000 pix, 70PCC	0.5599245	0.08376936
Polling 1000 pix, 70PCC	0.4711389	0.06046465
Missed Measurements 1000 pix, 70PCC	0.5328009	0.07581481
PPAE 1000 pix, 70PCC	0.5840359	0.08046465
Even Weights 1000 pix, 70PCC	0.4688431	0.06496801
Optimized 1000 pix , 70PCC, C1= 0.198276 C2= 0.002923, C3 = 0.384756, C4 = 0.414045	0.454877	0.060337
Equal Dispersion1000 pix, 95PCC	0.4170219	0.08348148
Polling 1000 pix, 95PCC	0.3454535	0.05293098
Missed Measurements 1000 pix, 95PCC	0.3840236	0.07993939
PPAE 1000 pix, 95PCC	0.4296922	0.08354545
Even Weights 1000 pix, 95PCC	0.3264846	0.06317677
Optimized 1000 pix , 95PCC, C1= 0.283978, C2= 0.264342, C3 = 0.339912, C4 = 0.111768	0.328182	0.051384
ALL HSI, 70PCC	0.2358412	0.03206061
ALL HSI, 95PCC	0.01801731	0.05822054

Table 22 - 1 Frame/Sec ; Noise = 10

1FPS,10 noise, 10 Min	IDE	FMT
No HSI	0.9647488	0.1416902
Equal Dispersion10 pix, 70PCC	0.9642417	0.1418300
Polling 10 pix, 70PCC	0.9602860	0.1407525
Missed Measurements 10 pix, 70PCC	0.9626051	0.1418721
PPAE 10 pix, 70PCC	0.9599452	0.1384613
Even Weights 10 pix, 70PCC	0.9635186	0.1427323
Optimized 10 pix , 70PCC, C1= 0.115270, C2= 0.227741, C3 = 0.163441, C4 = 0.493548	0.949767	0.128899
Equal Dispersion10 pix, 95PCC	0.9632313	0.1414731
Polling 10 pix, 95PCC	0.9571508	0.1374731
Missed Measurements 10 pix, 95PCC	0.9616853	0.1397879
PPAE 10 pix, 95PCC	0.9664639	0.1439394
Even Weights 10 pix, 95PCC	0.9567859	0.1362609
Optimized 10 pix , 95PCC, C1= 0.151558, C2= 0.346349, C3 = 0.107459, C4 = 394634	0.944548	0.128899
Equal Dispersion100 pix, 70PCC	0.9498382	0.1405606
Polling 100 pix, 70PCC	0.9457615	0.1377424
Missed Measurements 100 pix, 70PCC	0.9500531	0.1388569
PPAE 100 pix, 70PCC	0.9429607	0.1291684
Even Weights 100 pix, 70PCC	0.9508134	0.1373754
Optimized 100 pix , 70PCC, C1= 0.204441, C2= 0.208773 C3 = 0.099716, C4 = 0.487070	0.936878	0.127981
Equal Dispersion100 pix, 95PCC	0.9417355	0.1353552
Polling 100 pix, 95PCC	0.9407059	0.1389428
Missed Measurements 100 pix, 95PCC	0.9422240	0.1383401
PPAE 100 pix, 95PCC	0.9398181	0.1347138
Even Weights 100 pix, 95PCC	0.9412087	0.1362189
Optimized 100 pix , 95PCC, C1= 0.119903 C2= 0.133143, C3 = 0.328117, C4 = 0.418837	0.933010	0.127709
Equal Dispersion1000 pix, 70PCC	0.8441942	0.1206111

Polling 1000 pix, 70PCC	0.8291099	0.1150791
Missed Measurements 1000 pix, 70PCC	0.8408689	0.1200808
PPAE 1000 pix, 70PCC	0.8582939	0.1169226
Even Weights 1000 pix, 70PCC	0.8291622	0.1129259
Optimized 1000 pix , 70PCC, C1= 0.017380, C2= 0.247372, C3 = 0.158304, C4 = 0.576944	0.818109	0.105700
Equal Dispersion1000 pix, 95PCC	0.7858719	0.1202340
Polling 1000 pix, 95PCC	0.7591986	0.1130976
Missed Measurements 1000 pix, 95PCC	0.7732720	0.1179495
PPAE 1000 pix, 95PCC	0.7923889	0.1145741
Even Weights 1000 pix, 95PCC	0.7540341	0.1075758
Optimized 1000 pix , 95PCC, C1= 0.012228 C2= 0.329843, C3 = 0.479088, C4 = 0.178841	0.745940	0.101764
ALL HSI, 70PCC	0.2789145	0.02693603
ALL HSI, 95PCC	0.02301062	0.06137542

Table 23 - 30 Frames/Sec ; Noise = 3

30FPS,3 noise, 10 Min	IDE	FMT
No HSI	0.7697684	0.1735797
Equal Dispersion10 pix, 70PCC	0.7416232	0.1680277
Polling 10 pix, 70PCC	0.6943142	0.1564958
Missed Measurements 10 pix, 70PCC	0.7390102	0.1661506
PPAE 10 pix, 70PCC	0.6549051	0.1419744
Even Weights 10 pix, 70PCC	0.6274965	0.1351892
Optimized 10 pix , 70PCC, C1= 0.014627, C2= 0.112932, C3 = 0.500390, C4 = 0.372051	0.616789	0.134144
Equal Dispersion10 pix, 95PCC	0.7321851	0.1670817
Polling 10 pix, 95PCC	0.6651478	0.1564870
Missed Measurements 10 pix, 95PCC	0.7321053	0.1678113
PPAE 10 pix, 95PCC	0.6265312	0.1467127
Even Weights 10 pix, 95PCC	0.5908999	0.1388854
Optimized 10 pix , 95PCC, C1= 0.018418, C2= 0.294225, C3 = 0.353573, C4 = 0.333784	0.569240	0.134768
Equal Dispersion100 pix, 70PCC	0.6567437	0.1518332
Polling 100 pix, 70PCC	0.4683686	0.1170745
Missed Measurements 100 pix, 70PCC	0.6468613	0.1495918
PPAE 100 pix, 70PCC	0.5479848	0.1239770
Even Weights 100 pix, 70PCC	0.4559069	0.1116670
Optimized 100 pix , 70PCC, C1= 0.004887, C2= 0.134236, C3 = 0.456902, C4 = 0.403975	0.386778	0.101072
Equal Dispersion100 pix, 95PCC	0.6004164	0.1514802
Polling 100 pix, 95PCC	0.3716509	0.1107591
Missed Measurements 100 pix, 95PCC	0.6004059	0.1537077
PPAE 100 pix, 95PCC	0.4746321	0.1265252
Even Weights 100 pix, 95PCC	0.3626660	0.1087614
Optimized 100 pix , 95PCC, C1= 0.012373, C2= 0.303547, C3 = 0.424716, C4 = 0.259364	0.293827	0.096749
Equal Dispersion1000 pix, 70PCC	0.2862732	0.09570312

Polling 1000 pix, 70PCC	0.1685148	0.08091336
Missed Measurements 1000 pix, 70PCC	0.2814434	0.09836407
PPAE 1000 pix, 70PCC	0.2639776	0.09310481
Even Weights 1000 pix, 70PCC	0.1640531	0.08666489
Optimized 1000 pix , 70PCC; C1= 0.359479, C2= 0.001792, C3 = 0.397545, C4 = 0.241184	0.160592	0.081619
Equal Dispersion1000 pix, 95PCC	0.1114010	0.09273747
Polling 1000 pix, 95PCC	0.09925037	0.07870029
Missed Measurements 1000 pix, 95PCC	0.1060723	0.09697177
PPAE 1000 pix, 95PCC	0.06903472	0.09179677
Even Weights 1000 pix, 95PCC	0.06115937	0.08633244
Optimized 1000 pix , 95PCC; C1= 0.333142, C2= 0.139106, C3 = 0.268896, C4 = 0.258856	0.052338	0.081618
ALL HSI, 70PCC	0.1506782	0.06988402
ALL HSI, 95PCC	0.005552042	0.07113254

Table 24 - 30 Frames/Sec ; Noise = 10

30FPS,10 noise, 10 Min	IDE	FMT
No HSI	0.9784810	0.2276735
Equal Dispersion10 pix, 70PCC	0.9740640	0.2281936
Polling 10 pix, 70PCC	0.9696824	0.2266788
Missed Measurements 10 pix, 70PCC	0.9706234	0.2265345
PPAE 10 pix, 70PCC	0.9225203	0.1984168
Even Weights 10 pix, 70PCC	0.9253499	0.2015233
Optimized 10 pix , 70PCC; C1= 0.227027, C2= 0.367442, C3 = 0.295068, C4 = 0.110463	0.917342	0.195146
Equal Dispersion10 pix, 95PCC	0.9670640	0.2274208
Polling 10 pix, 95PCC	0.9587382	0.2271658
Missed Measurements 10 pix, 95PCC	0.9652804	0.2270362
PPAE 10 pix, 95PCC	0.9090186	0.1992848
Even Weights 10 pix, 95PCC	0.9099361	0.2001861
Optimized 10 pix , 95PCC; C1= 0.231871, C2= 0.245440, C3 = 0.304107, C4 = 0.218582	0.902617	0.196937
Equal Dispersion100 pix, 70PCC	0.8868362	0.1984563
Polling 100 pix, 70PCC	0.8638009	0.1924300
Missed Measurements 100 pix, 70PCC	0.8864160	0.1982439
PPAE 100 pix, 70PCC	0.8257688	0.1551459
Even Weights 100 pix, 70PCC	0.8068882	0.1516722
Optimized 100 pix , 70PCC; C1= 0.014147 C2= 0.249767, C3 = 0.474669, C4 = 0.261417	0.795062	0.147896
Equal Dispersion100 pix, 95PCC	0.8535518	0.2034804
Polling 100 pix, 95PCC	0.8146717	0.1962504
Missed Measurements 100 pix, 95PCC	0.8559756	0.2039259
PPAE 100 pix, 95PCC	0.7717705	0.1606551
Even Weights 100 pix, 95PCC	0.7359678	0.1561389
Optimized 100 pix , 95PCC; C1= 0.001742 C2= 0.158523, C3 = 0.319101, C4 = 0.520634	0.718417	0.150345
Equal Dispersion1000 pix, 70PCC	0.4896108	0.09712921

Polling 1000 pix, 70PCC	0.4884084	0.09703201
Missed Measurements 1000 pix, 70PCC	0.4903798	0.09884161
PPAE 1000 pix, 70PCC	0.6521430	0.1176792
Even Weights 1000 pix, 70PCC	0.4895740	0.09726531
Optimized 1000 pix , 70PCC; C1= 0.323041 C2= 0.000171, C3 = 0.032094, C4 = 0.64694	0.482933	0.096565
Equal Dispersion1000 pix, 95PCC	0.3098009	0.09566217
Polling 1000 pix, 95PCC	0.3198584	0.09485940
Missed Measurements 1000 pix, 95PCC	0.3096594	0.09689441
PPAE 1000 pix, 95PCC	0.4657255	0.1196411
Even Weights 1000 pix, 95PCC	0.3111236	0.09522858
Optimized 1000 pix , 95PCC; C1= 0.789575, C2= 0.000018, C3 = 0.012607, C4 = 0.1978	0.303919	0.095072
ALL HSI, 70PCC	0.2072435	0.07128082
ALL HSI, 95PCC	0.007360053	0.07468206

Table 25 - Utility Component Distance to Perfection and Group Difference in Distance; 1 Frame/Sec ; Noise = 3

1FPS, 3 noise, 10 Min	Distance to (0,0)	Group Difference in Distance
No HSI	0.89593514	
Equal Dispersion 10 pix, 70PCC	0.89132456	0.016006
Polling 10 pix, 70PCC	0.883763428	0.008445
Missed Measurements 10 pix, 70PCC	0.891589137	0.016271
PPAE 10 pix, 70PCC	0.875318565	0
Even Weights 10 pix, 70PCC	0.878247244	0.002929
Optimized 10 pix , 70PCC	0.867565914	-0.007752651
Equal Dispersion 10 pix, 95PCC	0.881418592	0.013679
Polling 10 pix, 95PCC	0.882801714	0.015062
Missed Measurements 10 pix, 95PCC	0.886793436	0.019054
PPAE 10 pix, 95PCC	0.872608129	0.004868
Even Weights 10 pix, 95PCC	0.867739716	0
Optimized 10 pix , 95PCC	0.862287775	-0.005451941
Equal Dispersion 100 pix, 70PCC	0.84852927	0.042956
Polling 100 pix, 70PCC	0.822838628	0.017265
Missed Measurements 100 pix, 70PCC	0.841151211	0.035578
PPAE 100 pix, 70PCC	0.827159154	0.021586
Even Weights 100 pix, 70PCC	0.80557355	0
Optimized 100 pix , 70PCC	0.793472277	-0.012101272
Equal Dispersion100 pix, 95PCC	0.82830138	0.06199
Polling 100 pix, 95PCC	0.776446712	0.010135
Missed Measurements 100 pix, 95PCC	0.818940713	0.052629
PPAE 100 pix, 95PCC	0.793763297	0.027452

Even Weights 100 pix, 95PCC	0.766311292	0
Optimized 100 pix , 95PCC	0.748978879	-0.017332413
Equal Dispersion1000 pix, 70PCC	0.566156119	0.092833
Polling 1000 pix, 70PCC	0.475002986	0.00168
Missed Measurements 1000 pix, 70PCC	0.538167896	0.064845
PPAE 1000 pix, 70PCC	0.58955279	0.11623
Even Weights 1000 pix, 70PCC	0.473323034	0
Optimized 1000 pix , 70PCC	0.458861241	-0.014461793
Equal Dispersion1000 pix, 95PCC	0.425295688	0.092755
Polling 1000 pix, 95PCC	0.349485063	0.016944
Missed Measurements 1000 pix, 95PCC	0.392255569	0.059715
PPAE 1000 pix, 95PCC	0.437738768	0.105198
Even Weights 1000 pix, 95PCC	0.332540972	0
Optimized 1000 pix , 95PCC	0.332180283	-0.000360689
ALL HSI, 70PCC	0.238010408	
ALL HSI, 95PCC	0.060944686	

Table 26 - Utility Component Distance to Perfection and Group Difference in Distance; 1 Frame/Sec ; Noise = 10

1FPS,10 noise, 10 Min	Distance to (0,0)	Group Difference in Distance
No HSI	0.975098128	
Equal Dispersion10 pix, 70PCC	0.974616748	0.004737
Polling 10 pix, 70PCC	0.970546479	0.000667
Missed Measurements 10 pix, 70PCC	0.973003737	0.003124
PPAE 10 pix, 70PCC	0.969879538	0
Even Weights 10 pix, 70PCC	0.974033163	0.004154
Optimized 10 pix , 70PCC	0.958473947	-0.011405591
Equal Dispersion10 pix, 95PCC	0.973565188	0.007125
Polling 10 pix, 95PCC	0.966972858	0.000533
Missed Measurements 10 pix, 95PCC	0.971791785	0.005352
PPAE 10 pix, 95PCC	0.977123851	0.010684
Even Weights 10 pix, 95PCC	0.966440009	0
Optimized 10 pix , 95PCC	0.953302615	-0.013137394
Equal Dispersion100 pix, 70PCC	0.960182216	0.008416
Polling 100 pix, 70PCC	0.955739391	0.003973
Missed Measurements 100 pix, 70PCC	0.960146932	0.00838
PPAE 100 pix, 70PCC	0.951766441	0
Even Weights 100 pix, 70PCC	0.960686277	0.00892
Optimized 100 pix , 70PCC	0.945578935	-0.006187505
Equal Dispersion100 pix, 95PCC	0.951413045	0.001989
Polling 100 pix, 95PCC	0.950911506	0.001488
Missed Measurements 100 pix, 95PCC	0.952325601	0.002902
PPAE 100 pix, 95PCC	0.949423967	0
Even Weights 100 pix, 95PCC	0.951014935	0.001591

Optimized 100 pix , 95PCC	0.941709748	-0.007714219
Equal Dispersion1000 pix, 70PCC	0.852766606	0.01595
Polling 1000 pix, 70PCC	0.837058197	0.000241
Missed Measurements 1000 pix, 70PCC	0.849399732	0.012583
PPAE 1000 pix, 70PCC	0.866221284	0.029405
Even Weights 1000 pix, 70PCC	0.836816714	0
Optimized 1000 pix , 70PCC	0.82490898	-0.011907734
Equal Dispersion1000 pix, 95PCC	0.795016263	0.033347
Polling 1000 pix, 95PCC	0.767576434	0.005907
Missed Measurements 1000 pix, 95PCC	0.782215872	0.020547
PPAE 1000 pix, 95PCC	0.800629373	0.03896
Even Weights 1000 pix, 95PCC	0.761669204	0
Optimized 1000 pix , 95PCC	0.752849517	-0.008819687
ALL HSI, 70PCC	0.280212148	
ALL HSI, 95PCC	0.065547165	

Table 27 - Utility Component Distance to Perfection and Group Difference in Distance; 30 Frames/Sec ; Noise = 3

30FPS,3 noise, 10 Min	Distance to (0,0)	Group Difference in Distance
No HSI	0.78909651	
Equal Dispersion10 pix, 70PCC	0.760419804	0.118526
Polling 10 pix, 70PCC	0.711732495	0.069838
Missed Measurements 10 pix, 70PCC	0.757457654	0.115564
PPAE 10 pix, 70PCC	0.670117468	0.028223
Even Weights 10 pix, 70PCC	0.641894055	0
Optimized 10 pix , 70PCC	0.631207797	-0.010686257
Equal Dispersion10 pix, 95PCC	0.751006868	0.144005
Polling 10 pix, 95PCC	0.683307966	0.076306
Missed Measurements 10 pix, 95PCC	0.751091741	0.144089
PPAE 10 pix, 95PCC	0.643479573	0.036477
Even Weights 10 pix, 95PCC	0.607002344	0
Optimized 10 pix , 95PCC	0.584975719	-0.022026626
Equal Dispersion100 pix, 70PCC	0.674066472	0.204683
Polling 100 pix, 70PCC	0.482779022	0.013396
Missed Measurements 100 pix, 70PCC	0.663933165	0.19455
PPAE 100 pix, 70PCC	0.561834173	0.092451
Even Weights 100 pix, 70PCC	0.469383234	0
Optimized 100 pix , 70PCC	0.399765895	-0.069617339
Equal Dispersion100 pix, 95PCC	0.619230252	0.240607
Polling 100 pix, 95PCC	0.387804035	0.009181
Missed Measurements 100 pix, 95PCC	0.619768749	0.241145
PPAE 100 pix, 95PCC	0.491206939	0.112584
Even Weights 100 pix, 95PCC	0.378623388	0

Optimized 100 pix , 95PCC	0.309345559	-0.069277829
Equal Dispersion1000 pix, 70PCC	0.301846703	0.116309
Polling 1000 pix, 70PCC	0.186933704	0.001396
Missed Measurements 1000 pix, 70PCC	0.298137347	0.1126
PPAE 1000 pix, 70PCC	0.279915485	0.094378
Even Weights 1000 pix, 70PCC	0.185537659	0
Optimized 1000 pix , 70PCC	0.180142864	-0.005394794
Equal Dispersion1000 pix, 95PCC	0.144949719	0.039149
Polling 1000 pix, 95PCC	0.126666379	0.020866
Missed Measurements 1000 pix, 95PCC	0.143717977	0.037917
PPAE 1000 pix, 95PCC	0.114858346	0.009058
Even Weights 1000 pix, 95PCC	0.105800561	0
Optimized 1000 pix , 95PCC	0.096957538	-0.008843023
ALL HSI, 70PCC	0.166095443	
ALL HSI, 95PCC	0.071348885	

Table 28 - Utility Component Distance to Perfection and Group Difference in Distance; 30 Frames/Sec ; Noise = 10

30FPS,10 noise, 10 Min	Distance to (0,0)	Group Difference in Distance
No HSI	1.004619475	
Equal Dispersion10 pix, 70PCC	1.000436402	0.056819
Polling 10 pix, 70PCC	0.995824902	0.052208
Missed Measurements 10 pix, 70PCC	0.996708415	0.053091
PPAE 10 pix, 70PCC	0.943616941	0
Even Weights 10 pix, 70PCC	0.947039639	0.003423
Optimized 10 pix , 70PCC	0.937869024	-0.00575
Equal Dispersion10 pix, 95PCC	0.993445016	0.062838
Polling 10 pix, 95PCC	0.985283328	0.054676
Missed Measurements 10 pix, 95PCC	0.991620737	0.061014
PPAE 10 pix, 95PCC	0.930606924	0
Even Weights 10 pix, 95PCC	0.931696399	0.001089
Optimized 10 pix , 95PCC	0.92385152	-0.006755404
Equal Dispersion100 pix, 70PCC	0.90877024	0.087751
Polling 100 pix, 70PCC	0.88497531	0.063956
Missed Measurements 100 pix, 70PCC	0.908313805	0.087294
PPAE 100 pix, 70PCC	0.840216854	0.019197
Even Weights 100 pix, 70PCC	0.821019503	0
Optimized 100 pix , 70PCC	0.808700693	-0.01231881
Equal Dispersion100 pix, 95PCC	0.877470768	0.125122
Polling 100 pix, 95PCC	0.837976252	0.085628
Missed Measurements 100 pix, 95PCC	0.879931816	0.127584
PPAE 100 pix, 95PCC	0.788314509	0.035966
Even Weights 100 pix, 95PCC	0.752348296	0

Optimized 100 pix , 95PCC	0.733979976	-0.01836832
Equal Dispersion1000 pix, 70PCC	0.4991521	0.001198
Polling 1000 pix, 70PCC	0.497953789	0
Missed Measurements 1000 pix, 70PCC	0.500241954	0.002288
PPAE 1000 pix, 70PCC	0.662675552	0.164722
Even Weights 1000 pix, 70PCC	0.499142507	0.001189
Optimized 1000 pix , 70PCC	0.492492722	-0.005461067
Equal Dispersion1000 pix, 95PCC	0.324234249	0
Polling 1000 pix, 95PCC	0.333628089	0.009394
Missed Measurements 1000 pix, 95PCC	0.324464899	0.000231
PPAE 1000 pix, 95PCC	0.480847413	0.156613
Even Weights 1000 pix, 95PCC	0.325371137	0.001137
Optimized 1000 pix , 95PCC	0.318442214	-0.005792035
ALL HSI, 70PCC	0.219159357	
ALL HSI, 95PCC	0.075043857	

REPORT DOCUMENTATION PAGE						*Form Approved* *OMB No. 074-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* 01-12-2010	**2. REPORT TYPE** **Doctoral Dissertation**	**3. DATES COVERED** *(From – To)* Sep 2005 – Dec 2010
4. TITLE AND SUBTITLE A Linear Combination of Heuristics Approach to Spatial Sampling Hyperspectral Data for Target Tracking		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Secrest, Barry R., Major, USAF		**5d. PROJECT NUMBER** 08ENG240
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765		**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/DEE/ENG/10-08
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) (MR. DEVERT WICKER) AFRL/RYAT BLDG 620, 2241 AVIONICS CIRCLE WPAFB, OH 45433-7333 (937-904-9871 devert.wicker@wpafb.af.mil)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the United States Government and is not subject to copyright protection in the United States.

14. ABSTRACT
Persistent surveillance of the battlespace results in better battlespace awareness which aids in obtaining air superiority, winning battles, and saving friendly lives. Although hyperspectral imagery (HSI) data has proven useful for discriminating targets, it presents many challenges as a useful tool in persistent surveillance. A new sensor under development has the potential of overcoming these challenges and transforming our persistent surveillance capability by providing HSI data for a limited number of pixels and grayscale video for the remainder. The challenge of exploiting this new sensor is determining where the HSI data in the sensor's field of view will be the most useful. The approach taken is to use a utility function with components of equal dispersion, periodic poling, missed measurements, and predictive probability of association error (PPAE). The relative importance or optimal weighting of the different types of TOI is accomplished by a genetic algorithm using a multi-objective problem formulation. Experiments show using the utility function with equal weighting results in superior target tracking compared to any individual component by itself, and that equal weighting is close to the optimal solution. The new sensor is successfully exploited resulting in improved persistent surveillance.

15. SUBJECT TERMS
Sensor Resource Manager, Target Tracking, Kalman Filter, Hyperspectral Image, Data Fusion, Panchromatic Video, Persistent Surveillance

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Mendenhall, Michael, Major, USAF
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U	UU	193	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 4614; e-mail: michael.mendenhall@afit.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18